The Thesis Committee for Gautam Vijay Salhotra
Certifies that this is the approved version of the following thesis:

MODEL-BASED CONTROLLER DESIGN

AND SIMULATION OF

A MARINE CHILLER

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Raul G. Longoria

Co-Supervisor:

Thomas M. Kiehne

Model-based Controller Design and Simulation of a Marine Chiller

by

Gautam Vijay Salhotra, B.Tech; M.Tech.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2012

"To think is easy. To act is hard. But the hardest thing in the world is to act in accordance with your thinking"

- Johann Wolfgang von Goethe

Acknowledgements

The completion of this thesis would not have been possible without the guidance, support, and assistance of the following individuals:

My family, based in Mumbai. Without your love, support and unwavering belief in my abilities, I would not be the man I am today.

My colleagues at The University of Texas, notably Felipe Lopez, Jonathan LeSage, and Vikram Devaraj, who were willing to provide assistance whenever needed.

My lifelong friends from India, who played a major part in shaping who I am today. I'm glad to have shared so many wonderful experiences with you.

My friends from Halstead co-op and Austin, who come from all walks of life, and various cultures across the US and the world. It is a real pleasure knowing all of you, and sharing thoughts, ideas and experiences.

My teachers and professors, who have guided me in life as well as education, notably Oswald Pereira, Shalini Dongre, and Dr S V Sreenivasan.

Dr Longoria, whose crucial insights and feedback helped direct the course of this thesis considerably. Your help in my times of doubt and uncertainty has been immensely helpful.

And finally, Dr Kiehne, who constantly provided encouragement and believed in me. It is your supervision, direction and inspiration that has provided me with the motivation to work harder and achieve this gigantic milestone in my life.

10th August 2012

Abstract

Model-based Controller Design and Simulation of a Marine Chiller

by

Gautam Vijay Salhotra, M.S.E.

The University of Texas at Austin, 2012

Co-Supervisor: Thomas M. Kiehne

Co-Supervisor: Raul G. Longoria

For the past decade, the US Navy has committed to fundamental research and technology development on its next generation of surface ships. The vision is that these warships will be dynamically reconfigurable, energy-efficient, and have state-of-the-art pulsed energy weapons and sensors onboard. These developments represent a significant increase in highly dynamic on-board electrical systems that will produce correspondingly large amounts of dynamic heat generation, which, if not managed properly, will likely produce significant thermal side effects.

In previous work, a highly customizable simulation framework has been developed to address thermal management issues across both the mechanical and electrical domains. This software environment is called the Dynamic Thermal Modeling and Simulation (*DTMS*) framework. The purpose of the current work is to introduce

modern control theory into *DTMS*, thus providing the framework with the ability to control large-scale system simulations.

The research reported in this thesis uses control of a marine chiller as a simulation vehicle. Several control strategies were implemented. These included the well-established PID controller as well as a new controller based on optimal control theory. Results for chiller simulations in the case of no-control, PID control, and optimal control are presented here. The comparative effectiveness of these controls in bringing the chiller to startup equilibrium is investigated. Response of the chiller model and the optimal controller to highly dynamic, varying heat loads was tested.

The PID controller in *DTMS* is modeled as a special case of the transfer function control scheme. A PID controller is simple to implement but responses are inherently local and multiple controls in a system or subsystem simulation can easily lead to conflicts. The optimal control problem has been modeled as an Infinite Horizon Linear Quadratic Regulator (LQR) problem. This formulation is not local and does not create undesirable effects in parts of the system that not controlled directly by controller inputs.

Using the York 200-ton marine chiller as an example, specific steps required to formulate the LQR problem are documented in this report. Implementation of the LQR controller was demonstrated for the startup to steady-state function of the chiller at full load. Treatment of the optimal controller ends with simulation of the chiller and its LQR controller under the influence of varying dynamic heat loads in a chilled water loop. The heat load variation examined has highly transient characteristics that affect the temperature of the fresh water entering the chiller, as well as the refrigerant pressure and temperature in the evaporator. The LQR formulation is shown to actively adjust to these varying operating points in a smooth and responsive manner.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Recent events in the international geo-political arena have seen a significant shift in international security perceptions and global tensions. This has led various nations to upgrade their defensive and offensive posture, subject only to constraints on available vital natural resources. This political situation, and the growth of 3rd world economies, has resulted in a marked increase in global fossil fuel procurement and consumption, which has only increased the scarcity of these fuels, while creating concerns about increased "greenhouse" gas emissions and global warming. These conditions have brought about a push for increased fuel efficiency and alternative, sustainable energy sources. In addition, widespread international terrorism, the threat of nuclear proliferation, and unstable governments create perverse and often baffling scenarios in geopolitics, leading to international tensions between countries. This has focused attention on more modular and versatile defense systems with an emphasis on a global presence. A country recovering from a financial crisis has additionally put increased pressure on efficiency in defense expenditures, which may face a reduction subject to the evolution of the American economy.

In light of these geopolitical, environmental, and economic considerations, research is ongoing to develop the next generation of weaponry and power projection. In particular, the United States Navy is undertaking projects to develop modern defense systems including a maritime fleet that is efficient, flexible, and adaptable to a variety of environmental conditions. Specifically, the Navy has committed to the use of electrical

energy as a primary source of energy transport on its future ships [1]. This thesis gives particular attention to the work in improving thermal management onboard these ships. This chapter discusses the motivation behind the Navy's decision to pursue the development of an All-Electric Ship (AES), as well as the necessity for improved and advanced thermal management onboard these ships. In addition, this chapter provides a synopsis of work presented in this thesis report.

1.1 THE ALL-ELECTRIC SHIP (AES)

1.1.1 Motivation

The next generation of Navy ships must be more reconfigurable, energy-efficient, and versatile than their present day counterparts. To be viable, these ships should possess certain qualities over and above the features present in the ships of today. First and foremost, future ships need to be reconfigurable, so as to give them the versatility to respond to the dynamic nature of maritime combat and present broad offensive capabilities when confronted by various adversaries. Secondly, efficiency is of paramount importance to reduce fuel and power related costs which, when combined with reconfigurability, facilitate harnessing large amounts of power in short periods of time as may be required during combat. These factors are collectively responsible for the Navy's decision to pursue development of an AES.

The current fleet of naval surface ships, such as the *Arleigh Burke* DDG-51 class destroyer, uses different prime energy sources for the propulsion and onboard equipment.

The DDG-51 employs separate gas turbines for each of these facets of energy production, which then typically run at partial load, leading to large inefficiencies and poor utilization of available energy [2]. Central to the vision of a future AES fleet is the conversion of mechanical prime power to electric power via synchronous generation. This generated power is then distributed via a distribution grid to ship propulsion motors and also to onboard equipment. This configuration for power distribution is called an Integrated Power System (IPS).

There are also other motivations for the shift to an AES fleet. Connection of propellers to the motor drives removes intermediate fuel inefficient propulsion components such as gearboxes and their undesirable speed-shift dynamics. This streamlining of energy transport from the distribution grid to motors to the propellers increases efficiency merely by removing these components and reducing the amount of lost power in achieving propulsion. There is also a desire for employment of advanced, high-energy weapons in the next generation of ships. These might include the electro-magnetic rail gun, free-electron laser, and other high-energy weapons. All of these weapons require significantly larger amounts of energy and may even rival propulsion as the most power hungry component on the ship. When these weapon systems are employed, automatic reconfigurability of the shipboard power distribution will be a desired feature such that available power can be used to its fullest.

In short, the AES should be able to provide a variety of new features to the current fleet. It needs to have an adaptively reconfigurable distribution grid so as to provide energy as and when required by any component while still running motor drives

and electricity generation at maximum efficiency. It needs to be able to dynamically harness large amounts of energy required in the face of various events such as combat or unfavorable environmental conditions. An IPS and a reconfigurable power grid achieve these goals via all-electric power distribution and consumption.

1.1.2 Development of the ESRDC

In 2002, the Office of Naval Research (ONR) funded the creation of a consortium comprised of various universities involved in aspects of AES research. This organization was dubbed the Electric Ship Research and Development Consortium (ESRDC) and given the task of conducting research for the development of systems and technologies necessary for the creation of an AES. Over time the ESRDC expanded from its original membership of four universities to a current configuration involving eight universities: the University of Texas at Austin, Florida State University, Massachusetts Institute of Technology, Purdue University, Mississippi State University, the University of South Carolina, the United States Naval Academy, and the Naval Post Graduate School. June 2012 marked the tenth anniversary of the ESRDC, acknowledging a decade of multidisciplinary research that focused on system complexity and the development of tools for simulation and system design of various aspects of an AES, with an objective of reducing experimental costs and risks associated with early design decisions [3].

Throughout its first 10 years, the ESRDC acknowledged the need for commercial software packages as well as in-house simulation software to provide flexible and detailed analytical tools to facilitate decision making in the design process. Harnessing

the power of detailed simulations promises reduced design and build costs during the early and detailed design process, thus making them a highly cost-effective alternative during AES design. Recent efforts by the ESRDC have relied increasingly on in-house simulation software to provide the developer and designer with increased flexibility and a more in-depth and versatile developmental environment for system-level modeling. Development of flexible, accessible, dynamic software that includes aspects of system controls while incorporating electrical, mechanical, and thermal interactions for design and optimization remains a key element of the on-going role of the ESRDC.

In order to acquire the science and technology base needed by the Navy for the development of the next-generation energy efficient AES, the ESRDC organized it's computational research into five main categories: computational tools for early ship design, ship electric power system, total ship system solution to thermal management, load management, and next generation integrated power system (NGIPS) [4]. More specifically, in the context of this thesis, a sophisticated framework has been developed using the object-oriented C++ environment to simulate thermal-electrical-mechanical interactions under highly transient conditions, which will likely be the case for an AES during combat. This software environment is called the Dynamic Thermal Modeling and Simulation (*DTMS*) framework.

1.2 THERMAL MANAGEMENT AND *DTMS*

1.2.1 The case for Thermal Management Research for the All-Electric Ship (AES)

The next-generation AES is expected to employ a large number of new and advanced technologies that will radically change the propulsion, weapon, and sensor systems onboard. Some of these systems will require large amounts of energy, potentially on very short notice in the case of combat. These large, highly transient energy requirements will result in correspondingly large amounts of waste heat generation, which, if not properly managed, will have disastrous consequences for the AES and may put the lives of those onboard at risk [2].

As an essential measure to avoid these risks, the study of dynamic aspects of future thermal management systems is a critical element of the ESRDC's research for the AES. Effective thermal management is an essential facilitator on the AES that will enable the simultaneous use of various defense and propulsion systems while avoiding excessive heat generation and catastrophic component failure. With these thoughts in mind, the ESRDC has sought to create simulation tools to enable the ship designer to make informed decisions, from an understanding of the impact of dynamic thermal loads, while designing the AES. This approach is also beneficial from the standpoint of cost-effectiveness, as these simulations can replace experimental tests that would cost significantly larger amounts of money.

1.2.3 Commercial Software Packages versus In-House Software

When the ESRDC was formed in 2002, the participating universities understood the need for advanced modeling techniques to accurately simulate various components on the AES and their thermal-mechanical-electrical interactions. At that time there were few simulation tools available to model the behavior of the new technologies contemplated for the AES and their response to high-energy transients. There was a clear need to model complex electrical distribution grids that were dynamically reconfigurable, high-density energy storage devices, intense pulsed-load weapon and sensor systems, and adaptive thermal management techniques. With such a wide range of modeling challenges, each participating university quickly turned to a combination of commercial and in-house software to address specific needs in their respective research areas.

The thermal management research group at the University of Texas at Austin (UT) initially focused its attention on two commercial tools for thermal modeling and simulation. Steady-state representations of various components were modeled using *CycleTempo*, a thermal modeling framework developed and supported by Delft University of Technology in the Netherlands [25]. This modeling environment was specifically designed to facilitate system-level analyses for the management of power and refrigeration/cooling systems. Detailed models of various devices were developed using *CycleTempo*. These included a hybrid gas-turbine engine [2], a solid oxide fuel cell [5], and a 200-ton marine chiller model [6]. Dynamic simulations were performed using *ProTRAX*, a commercial modeling framework developed by TRAX International [26], which used FORTRAN as its programming base. It used a flow-effort modeling construct

similar to what is currently used in the *DTMS* framework developed by the thermal management team at UT. The *ProTRAX* environment provided the capability to model indefinitely large thermodynamic systems while employing robust, tunable feedback controls, thereby facilitating the generation of sophisticated simulations. Using *ProTRAX*, UT graduate students created dynamic models of shipboard pulsed weaponry [7], a 200-ton York marine chiller [6], and a notional Integrated Propulsion System [2] patterned after the DDG-1000.

These modeling tools were immensely useful to the AES thermal management research efforts at UT. However, these software packages, being proprietary in nature, were opaque in their source code – meaning that the user did not have complete control over all the variables and methods used in the devices being simulated. These software packages were tailored for specific applications and are not amenable to a multidisciplinary modeling approach that caters to the dynamic modeling needs of various electrical, mechanical, and thermal systems interacting with one another under extreme conditions. These limitations make such commercial packages less flexible for the modeler, especially when interacting devices with highly transient affects and features are to be modeled. This lack of versatility in modeling all possible functional states of a device, coupled with the high cost of purchase and maintenance of the software, the lack of portability from one energy medium to another, and minimal customization opportunities limits their usability for AES simulations. Thus, in recent years, the focus of the UT thermal management team has shifted from adapting commercial software products to development of an in-house, highly customizable, simulation environment

that has been specifically built for the problem at hand and is easily transportable to other users.

In December 2007, UT graduate student Patrick Paullus created the first functional version of such an in-house framework [8], later named Dynamic Thermal Modeling and Simulation (*DTMS*), specifically to address the thermal management modeling needs of the ESRDC while also maintaining portability across various platforms and research areas. In December 2009, UT graduate student Michael Pierce restructured the framework and improved on its ease of use for the modeller and end user [24]. He also created a very detailed system of debugging to facilitate error-identification in *DTMS*.

Designed from the ground up to be highly customizable and universally applicable, *DTMS* has become the primary development vehicle for all thermal management work currently ongoing at UT. Numerous models and several large-scale simulations have been developed using *DTMS.* Included among these are a simulation of the entire starboard freshwater chilling loop of the *Arliegh Burke* DDG-51 guided missile destroyer that uses a highly evolved dynamic chiller model with a PID (proportional-integral-differential) control system [9] and a thermo-electric co-simulation of the zonal electrical distribution system that makes up the Naval Combat Survivability Testbed [10].

Since its initial formulation in 2007, the *DTMS* framework has been used by several UT graduate students and has undergone numerous improvements and updates to make the framework more robust, stable, and versatile as well as adding the capability to handle various thermodynamic media. These models and simulations are documented in

the student reports and papers contained in the reference list. *DTMS* has evolved into a relatively sophisticated tool that is moderately easy to use for simulating various complex physical systems. However, the implementation of a controls framework in *DTMS* has not been advanced beyond a few basic formulations, currently including only simple PID and on-off controls. These control strategies peg themselves to one variable and a reference point, and are used to calculate control outputs based purely on the error between the variable and the desired reference, thus making their applicability very localized in nature. Such localized PID control is incapable of easily taking into consideration the interactions among various components during a dynamic simulation. As a result of these limitations, implementation of localized PID loops on a large, highly interconnected, and complex system such as a vapor compression chiller and its attendant loads may produce outputs that are often baffling to an end user who may not always understand the physics behind the complex interactions between components.

This thesis seeks to bridge the gap between the universality of *DTMS* and its current controls framework. A new and different approach to controlling a chiller has been introduced and formulated. This strategy, broadly known as Optimal Control Theory, employs the state equations of the system as a whole, thus taking into consideration all components and their resulting interactions using a more holistic approach. The principal advantage of this approach lies in calculating control outputs by quantifying changes in various parameters due to inter-component interactions as well as component-only physics. In this thesis Optimal Control Theory, which is based on linear algebra and involves linearizing a given system about an equilibrium point, has been

studied, formulated, and implemented on a marine chiller simulated entirely within *DTMS*. The approach has provided favorable results even for a highly non-linearized system such as this. Thus, in this thesis, the hypothesis that a non-linear system such as a vapor compression chiller can be controlled by a linearized control strategy using optimal controls will be addressed and proven to be valid.

1.3 ORGANIZATION OF THESIS

This thesis report addressed the following topics during the course of research conducted using *DTMS* for thermal management control:

- Simulation of a marine chiller in *DTMS* including the proofing of its various components (Chap 2),
- Functional changes incorporated into *DTMS* to make it more easily usable by the developer and end user (Chap 3),
- PID control strategy and its implementation on a stand-alone chiller (Chap 4),
- Introduction to the fundamentals of Optimal Control Theory and its implementation for the stand-alone chiller modeled in Chap 2 (Chap 5),
- Concluding chapter with a summary of results and remarks on the topic of controlling a non-linear system with a linear control strategy (Chap 6).

Chapter 2: *DTMS* Chiller Simulation

A marine chiller, similar in construct to the York 200-ton unit on the *Arleigh-Burke* class destroyer, was selected for development of the optimal control strategy in *DTMS*. During the learning process, an effort was made to understand the dynamics of the chiller by developing models for each of its components and then the system in its entirety. This enabled the developer to conveniently understand each model, the interactions between various models, and the equations employed in a simulation.

This chapter discusses the modeling of a generic chiller. Presented first is relevant literature addressing the modeling of chiller components and then the chiller as a whole. This is followed by a description of the model for each component and of the simulations used to test these models. Understanding interactions between components was tested by placing them in series in an open circuit, an investigation which culminated in an open-loop chiller simulation. Finally, each element was connected to form a model of an actual closed loop marine chiller with parameters patterned after the York 200-ton unit.

2.1 Literature Review

Most modern-day chillers follow the basic vapor-compression cycle, with the refrigerant fluid exchanging heat at an evaporator and condenser, gaining energy at the compressor to exit at a higher pressure and temperature, and then being expanded to a lower pressure in an expansion valve. A significant amount of research and engineering has been conducted on the behavior of these chillers to include their steady-state and

dynamic response under transient conditions. Most notably, for the purposes of this work, abundant literature is available which demonstrates that both the compressor and expansion valve have much faster response times than the phase change heat exchangers [11,12]. This occurs principally because there is significant inertia associated with heat and mass transfer in the condenser and evaporator, components which then play a dominant role in the overall dynamic response of the system. Therefore, in what follows, these elements are modeled dynamically while the compressor and thermostatic expansion valve are modeled statically. This approach is also highly beneficial to setup of the optimal control formulation presented in Chapter 5.

The literature on modeling of heat exchangers for the purpose of refrigeration is also extensive. Llopis et al. [13] discuss a control volume approach for modeling shell-and-tube evaporators with the refrigerant flowing in the tube and the secondary coolant in the shell of a shell-and-tube heat exchanger. In this approach, the tubes are divided into regions based on the state of the refrigerant flowing through them, i.e., whether the fluid is saturated or superheated. The saturated region is then further divided into discrete longitudinal control volumes representing the saturated liquid and saturated vapor. A schematic depicting this approach is shown in Figure 2.1 below.

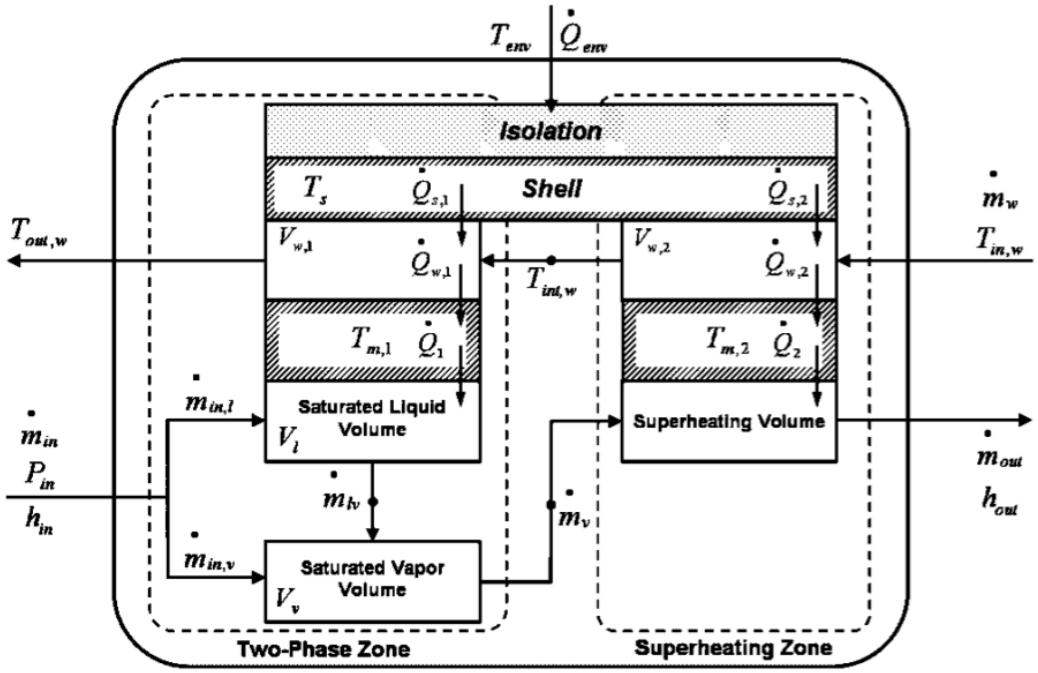

Figure 2.1: A shell-and-tube evaporator conceptual diagram, by Llopis [13]

A similar model for a shell-and-tube condenser was used in [14] where the refrigerant flow is through the shell. In the condenser, the control volumes for the refrigerant are a de-superheating zone, a condensing zone, and a sub-cooled zone, where the liquid is assumed to experience sub-cooling only by losing heat to the environment through the shell. Wang et al. [17] adopted a more detailed, spatially distributed approach for two-phase flow in the heat exchangers where parameters such as the void fraction were used to couple equations for mass flow and energy balance. Gruhle and Isermann [15] studied the dynamic behavior of the evaporator in a chiller using deterministic external inputs such as expansion valve position and disturbances such as air temperature or condenser pressure. Shah et al. [16] developed a model for subcritical vapor

compression systems with the option of multiple evaporators. Their simulations implemented model-based controls which was then a conceptually new approach for the air-conditioning industry. Zhang et al. [18] modeled a refrigerant cycle for cooling of electronics that employed the momentum equation, as well as mass and energy balance equations in micro-channels. The use of micro-channels in cooling of small scale electronic modules creates a significant pressure drop and demands the use of the momentum conservation equations for adequate representation of behavior. That demand is not seen in this study, nor is it desirable in light of the need to represent behaviors at the larger system-level.

2.2 MODEL DEVELOPMENT

This section considers the modeling of various components that make up a generic vapor compression chiller. The construct begins with basic components such as connecting pipes and orifice plates, and then moves on to more complicated devices such as the compressor and phase change heat exchangers. Default values for various parameters in each model are taken from the manual for the York 200-ton marine chiller [19]. These parameters are summarized in Section 2.3 that follows.

To take advantage of the object oriented nature of C++, each model is represented as a class in the *DTMS* framework and various instances of the class are connected to construct a circuit. Clearly, it is necessary to know the state of the fluid exiting each component and the relationships between energy exchanges, flow rates, and enthalpy.

Relevant equations will be provided where necessary. Additional details of this formulation may be found in references [8] and [9].

2.2.1 Pipe Model

The simple pipe model (class *Pipe*) in *DTMS* contains geometric and material parameters (length, cross sectional area, metal mass, etc.) for a given pipe. These parameters are associated with functions that set these values. *DTMS* also contains functions to calculate properties of the fluid in a pipe that are associated with physical processes such as pressure loss due to friction and/or heat exchange with the environment or with other pipes. These fluid functions form the basis for many physical models that are constructed using the pipe model; or, in the parlance of C++ programming, "child" classes of the class *Pipe*. For example, as will be seen in the next few subsections, the shell of a shell-and-tube heat exchanger is a child class of class *Pipe*, whereas the tube is a pipe itself.

2.2.2 Expansion Valve

The expansion valve (class *ExpansionValve*) in the York 200-ton chiller, and in many other chillers, is a simple orifice plate that is adjusted to control the flow rate of fluid from the high-pressure condenser to the low-pressure evaporator. This expansion valve is modeled such that the enthalpy of the fluid remains constant as it passes through the orifice plate from the high-pressure region to the low-pressure region. In a chiller, the valve position of a thermostatic expansion valve (TEV) determines the mass flow rate of

refrigerant through the circuit. Thus, the following simple relation equates the outlet enthalpy (h_{out}) to the inlet enthalpy (h_{in}):

$$\boldsymbol{h_{out} = h_{in}} \tag{2.1}$$

The expression for mass flow rate (W) through the valve is proportional to the square root of the pressure differential (ΔP), as follows:

$$\boldsymbol{W = C_f\sqrt{\Delta P}} \tag{2.2}$$

C_f is the flow coefficient which is determined by inlet density of the fluid (ρ_{in}), valve position (n_v), and also by design specifications such as the design flow rate (W_{des}), design inlet flow density ($\rho_{in,des}$), maximum pressure difference (ΔP_{max}) and the equilibrium valve position (n_{ve}).

$$\boldsymbol{C_f = \frac{n_v}{n_{ve}} * W_{des} * \sqrt{\frac{\rho_{in}}{\rho_{in,des} * \Delta P_{max}}}} \tag{2.3}$$

2.2.3 Centrifugal Compressor

The centrifugal compressor in *DTMS* (class *CentrifugalCompressor*) takes in saturated vapor refrigerant at low pressure and compresses it to a higher pressure. In the York 200-ton chiller, the compressor takes in refrigerant vapor at 0.323 MPa and compresses it to 1.021 MPa, to produce a compression ratio of 3.16:1. Ideally this compression is isentropic. However, a real compressor is not able to achieve this condition due to friction and other conditions that increase the entropy of the fluid. Thus,

an efficiency term is used to factor in the disparity between the isentropic exit enthalpy and the actual exit enthalpy of the compressed fluid.

The method used to calculate the mass flow rate through the compressor is based on the flow-pressure relationship described in [8]:

$$\boldsymbol{W = C_f\sqrt{\Delta P + s}} \tag{2.4}$$

Here, the flow coefficient depends on the vane position (p_v) and design vane position ($p_{v,des}$) according to:

$$\boldsymbol{C_f = W_{des} * \frac{p_v}{p_{v,des}} * \sqrt{\frac{\rho_{in}}{\rho_{in,des}}} * \frac{1}{\sqrt{\Delta P_{max} - \Delta P_{des}}}} \tag{2.5}$$

and

$$\boldsymbol{s = \Delta P_{max} * \frac{\rho_{in}}{\rho_{in,des}} * \frac{n_v}{n_{v,des}}} \tag{2.6}$$

The added term s in Equation 2.4 is called a source term, which arises due to the pressure head added by the compressor.

Simulation of the compression process is based on an overall isentropic efficiency. The outlet enthalpy of the fluid in the ideal, isentropic compression case is calculated using the outlet pressure (P_{out}) and inlet entropy (s_{in}), i.e.:

$$\boldsymbol{h_{isen} = h(P_{out}, s_{in})} \tag{2.7}$$

The compressor efficiency is then expressed as the ratio of the difference between the inlet and exit enthalpies in the case of ideal and the actual compression. Thus,

$$\frac{\boldsymbol{h_{isen} - h_{in}}}{\boldsymbol{h_{out} - h_{in}}} = \boldsymbol{\eta_{comp}} \tag{2.8}$$

and the resultant exit enthalpy of the compressed fluid is:

$$\boldsymbol{h_{out} = h_{in} + \frac{h_{isen} - h_{in}}{\eta_{comp}}} \tag{2.9}$$

Often, a compressor manufacturer provides a compressor map that is determined experimentally. This map plots normalized pressure against normalized volumetric flow rate. As discussed in previous *DTMS* compressor modeling efforts [9], certain efficiency and energy calculations use fluid properties, such as total displacement volume, that are not directly computed in *DTMS*.

In an effort to create a generic compressor model capable of modeling a large variety of centrifugal compressors, this work employs a generalized efficiency map for the purpose of calculating overall isentropic efficiency. The dimensionless head coefficient (Ω) and dimensionless flow coefficient (Θ) are sufficient to provide the overall efficiency. These coefficients depend on the impeller diameter (D_2) and the speed of sound in refrigerant vapor (a_1) as follows:

$$\boldsymbol{\Omega} = \frac{\boldsymbol{g_c \Delta H_{isen}}}{\boldsymbol{a_1^2}} = \frac{\boldsymbol{g_c}}{\boldsymbol{a_1^2}} (\boldsymbol{h_{isen} - h_{in}}) \tag{2.10}$$

$$\boldsymbol{\Theta} = \frac{\boldsymbol{W}}{\boldsymbol{a_1 D_1^2}} \tag{2.11}$$

If all terms on the right hand side of Equation (2.9) are in SI units, $g_c = 1$.

An additional feature of the above approach is that the calculation of efficiency is a stand-alone function in the compressor model. Therefore, if the user wishes to model another compressor, it is relatively easy to change the efficiency curve data inside the `calculateEfficiency()` function. The result of the efficiency calculation is then automatically updated using this new curve, thereby adding an element of flexibility and simplicity to the specifics of compressor representation in the model.

2.2.4 Single-Phase Heat Exchanger

A single-phase heat exchanger model was first created in *DTMS* as an intermediate step in modeling the condenser and evaporator of the chiller, both of which are two-phase heat exchangers. This model is a counter-flow, shell-and-tube heat exchanger with both shell and tube side represented as cylindrical pipes. The heat exchange between the shell-side pipe and tube-side pipes are subject to the thermal inertia of the pipe material surrounding the fluids flowing through them. This inertia must

be taken into account to adequately represent the rate of change of enthalpies for both the shell-side and tube-side fluids.

The NTU-effectiveness method was used to calculate heat exchange [23]. For a counter-flow heat exchanger, the maximum possible heat exchange (q_{max}) is expressed in terms of the heat capacity rate ($C=WC_p$), and the difference between the shell inlet temperature ($T_{sh,i}$) and the tube inlet temperature ($T_{tu,i}$):

$$\boldsymbol{q_{max} = C_{min}|T_{sh,i} - T_{tu,i}|} \tag{2.12}$$

Here C_{min} is the lesser of the heat capacity rates of the two fluids. The effectiveness (ϵ) associated with a heat exchanger in counter-flow operation is expressed as a function of the number of transfer units (NTU) and the heat capacity ratio ($C_r = C_{min}/C_{max}$):

$$\boldsymbol{\epsilon = \frac{1 - e^{-NTU(1-C_r)}}{1 - C_r e^{-NTU(1-C_r)}}} \tag{2.13}$$

Thus, the actual heat transfer (q_{actual}) between the shell and tube sides of the heat exchanger is a fraction of the maximum possible heat transfer, which is obtained by introducing the effectiveness as follows:

$$\boldsymbol{q_{actual} = q_{max} * \epsilon} \tag{2.14}$$

Simulations using this heat exchanger model are for fluids that remain in single-phase only. As in all *DTMS* models, fluid properties are updated using pressure and enthalpy. Therefore, the vapor-liquid state of the fluid does not factor into the calculation

of properties and, in practice, the single-phase and two-phase heat exchanger models are interchangeable. However, calculation of the heat exchanger effectiveness is based on the *NTU* and the heat capacity ratio of the fluids. The reader may recall that the heat capacity rate of a two-phase fluid is infinite since all of the heat exchange with the fluid during phase change is the latent heat of vaporization or condensation. Thus, $C_{max} \rightarrow \infty$ and $C_r = 0$ for two-phase heat exchange, and the effectiveness expression is greatly simplified. This circumstance, as well as other differences such as liquid levels in the shell side, requires a separate two-phase heat exchanger model and a separate model for the unique requirements of the shell-side of a counter-flow, shell-and-tube heat exchanger. This formulation is presented in the next section.

2.2.5 Two-Phase Shell Model

The two-phase shell model in *DTMS* (class *TPShellModel*) is modeled on a generic shell model (class *ShellModel*), which is a child class of class *Pipe*. Thus, in C++, *TPShellModel* inherits all of the parameters, functions, and features of the class *Pipe* and also has advanced features such as a saturated liquid reservoir, more complicated geometry, and the ability to handle multiple tubes each with multiple passes. These features are well suited to the York marine chiller, and can be readily used to construct a generic counter-flow, shell-and-tube heat exchanger with multiple tubes and multiple passes per tube.

In the York 200-ton condenser, the fluid reservoir serves as a refrigerant source for the thermostatic expansion valve connecting the condenser and evaporator. Also, the

geometry of the shell model considers the total surface area of the tubes (based on the number of tubes, their diameter, and the number of passes), and their arrangement inside the shell.

The rate of change of enthalpy in the fluid flowing through the shell [8] is determined by the rate of heat input (Q) and power transferred (P_t) and is represented as:

$$\frac{dh}{dt} = \frac{W_{in}(h_{in} - h_{out}) + Q - P_t}{m_m \frac{C_m}{C_f} + m_f} \tag{2.15}$$

Here, the denominator of the right hand side is the effective fluid mass and is the summation of the mass of the fluid (m_f) and the effective mass of fluid (m_{fi}) having the same heat capacity as that of the metal, i.e.:

$$\mathbf{m_{fi}} C_f = m_m C_m \tag{2.16}$$

Therefore

$$\mathbf{m_{fi}} = m_m \frac{C_m}{C_f} \tag{2.17}$$

After discretizing this equation over the time increment Δt, the resulting outlet enthalpy for the fluid at time $i+1$ is dependent on the enthalpy at time i according to:

$$h_{i+1} = \left(h_i - \frac{dh/dt}{\tau} \right) e^{-\tau \Delta t} + \frac{dh/dt}{\tau} \tag{2.18}$$

where τ is the time constant relating the mass flow rate of the fluid (W) and the effective fluid mass $\left(m_{eff} = \frac{m_m C_m}{C_f} + m_f \right)$ i.e.:

$$\boldsymbol{\tau = \frac{W}{m_{eff}} = \frac{W}{\frac{m_m C_m}{C_f} + m_f}} \tag{2.19}$$

To determine the mass flow rate in a two-phase shell model, first consider the single-phase flow dynamics in a simple pipe where the mass flow rate is determined from the square-root, flow-pressure relationship [8]:

$$\boldsymbol{W_{1\phi} = C\sqrt{\Delta P}} \tag{2.20}$$

in which the flow coefficient is:

$$\boldsymbol{C = \sqrt{\frac{\rho}{F_\mu (Af + B)}}} \tag{2.21}$$

F_μ is the correction factor for fluid viscosity (μ) and f is the friction factor. The scalar parameters A and B are defined in [21] as:

$$A = 2\frac{R_b N_{tcc}}{S_m^2}\left[R_L\left(N_b - 1\right) + 2\left(1 + \frac{N_{tcw}}{N_{tcc}}\right)\right] \tag{2.22}$$

$$B = \frac{N_b R_L\left(2 + 0.6\,N_{tcw}\right)}{2\,S_m S_w} \tag{2.23}$$

To extend this formulation into the realm of two-phase flow dynamics, the Lockhart-Martinelli correlation is used to relate the pressure drop across the single-phase

liquid-only flow to that across the two-phase liquid-vapor flow, with the help of a multiplicative coefficient. From Reference [22], where *x* is the vapor quality and L is the length of the pipe:

$$\left(\frac{\Delta P_f}{L}\right)_{2\phi} = \phi_L^2 \left(\frac{\Delta P_f}{L}\right)_L \tag{2.24}$$

and

$$\varphi_L^2 = 1 + \frac{20}{X_{tt}} + \frac{1}{X_{tt}^2} \tag{2.25}$$

$$X_{tt} = \left(\frac{1-x}{x}\right)^{0.9} \left(\frac{\rho_g}{\rho_f}\right)^{0.5} \left(\frac{\mu_f}{\mu_g}\right)^{0.1} \tag{2.26}$$

Thus, the two-phase flow-pressure correlation may be written as:

$$\boldsymbol{W_{2\phi} = \sqrt{\phi_L^2 \frac{\rho}{F_\mu(Af + B)}} \sqrt{\Delta P_L}} \tag{2.27}$$

This model may now be used to formulate the two-phase, shell-and-tube, counter-flow heat exchanger model described in the next sub-section.

2.2.6 Two-Phase Heat Exchanger

The two-phase heat exchanger (class *HXTwoPhase*) model is a significant advancement over the single-phase heat exchanger. It incorporates a more realistic shell

model that is capable of accounting for multiple tubes with multiple passes, as well as a liquid reservoir of saturated refrigerant liquid, and various tube arrangements inside the shell. This model may be used to simulate a wide variety of heat exchangers. However, there are limitations to its universality. This model is only applicable to a counter-flow, shell-and-tube heat exchanger. Also, the two-phase flow must occur in the shell, i.e., the refrigerant must flow through the shell and the secondary coolant must flow through the tubes. However, these limitations are not very restrictive since a developer may choose to create a model patterned after this one with the heat exchanger geometry of their choice.

As with the single-phase heat exchanger, the two-phase heat exchanger uses the *NTU*-effectiveness method to calculate heat transfer between the shell and the tubes. From well-established theory on this topic [23], the theoretically maximum possible heat transfer between the two fluids is again:

$$\boldsymbol{q_{max} = C_{min}|T_{sh,i} - T_{tu,i}|} \tag{2.28}$$

As for the single phase heat exchanger, the actual heat transfer is a fraction of the maximum heat transfer, as determined by the effectiveness, ε:

$$\boldsymbol{q_{actual} = q_{max} * \epsilon} \tag{2.29}$$

Again, the effectiveness is calculated as a function of the number of transfer units and the heat capacity ratio. However, in the case of two-phase flow, one of the fluids exchanging heat is undergoing a phase change. Thus, all the energy lost or gained by that fluid is in the form of latent heat. Therefore, the heat capacity of the phase change fluid

approaches infinity, its heat capacity is always C_{max}, and the effectiveness simplifies such that it is now purely an exponential function of *NTU*, i.e.:

$$\boldsymbol{\epsilon = 1 - e^{-NTU}} \tag{2.30}$$

Another important feature incorporated in the evaporator model is the change of evaporation pressure and temperature as a function of the heat load [32]. Consider the rate of change of vapor mass in the heat exchanger shell. Ensuring mass conservation, this is equal to the sum of the flow rate of the vapor entering the shell plus the vapor formed by evaporation of the liquid present in the shell minus the flow rate of vapor exiting the shell, i.e.:

$$\boldsymbol{\frac{dM_v}{dt} = W_{in}x_{in} + \frac{q_{actual}}{h_{lg}} - W_{out}} \tag{2.31}$$

Here, $W_{in}\,x_{in}$ is the fraction of vapor entering the shell and W_{out} is the flow rate of vapor exiting the shell. The rate of evaporation of refrigerant liquid is the rate at which heat is transferred to the shell (q_{actual}) relative to the latent heat of evaporation, i.e., the difference between the enthalpy of the saturated liquid and saturated vapor ($h_{lg}=h_l-h_g$). The left-hand side of this equation may be expressed as:

$$\boldsymbol{\frac{dM_v}{dt} = \frac{d}{dt}\left(V_v\rho_g\right) = V_v\frac{d\rho_g}{dP_e}\frac{dP_e}{dt}} \tag{2.32}$$

where the volume of vapor is assumed constant. Thus, dP_e/dt can be expressed as:

$$\frac{dP_e}{dt} = \frac{W_{in}x_{in} + \frac{q_{actual}}{h_{lg}} - W_{out}}{V_v \frac{d\rho_g}{dP_e}} \tag{2.33}$$

2.2.7 Shell-Side Exit Nodes

Most chiller manufacturers now design their heat exchangers in such a way that only the desired portion of the refrigerant exiting the heat exchanger goes to the next circuit component. Thus, the shell of a condenser acts as a reservoir from which only high-pressure, saturated liquid exits to the expansion valve. Likewise, in the evaporator, the refrigerant may often not be fully converted to low-pressure vapor. For these cases, the evaporator exit pipe is connected to the top of the shell, such that only saturated vapor exits to the compressor.

To mimic this behavior in a chiller simulation, shell exit nodes (classes *CondenserExitNode* and *EvaporatorExitNode*) are used to model the refrigerant exit condition on the shell-side of the condenser and evaporator. The function of these nodes is to update the fluid exit properties to saturated liquid or vapor, as the case may be, and provide this fluid state as an input to components downstream of the heat exchanger. This topic is discussed further in the simulations that follow.

2.3 CHILLER TEST SIMULATIONS

This section describes integration of the models described above to simulate a marine chiller. Component specifications and design parameters were taken from the York 200-ton chiller. These parameters are summarized in Table 2.1. The unit modeled is a Refrigerant-134a (R-134a) vapor-compression cycle. It employs chilled water as the evaporator secondary fluid and seawater as the condenser secondary fluid [19]. The subject chiller, along with its refrigerant flow diagram, is depicted in Figure 2.2.

Initially, simulations were conducted for an "open-loop chiller", i.e., fluid entered the chiller at the compressor inlet with a specified state and exited the chiller refrigerant circuit into a fluid reservoir. Therefore, the refrigerant always entered each component at a definite state, dependent only on the preceding components and the incoming fluid state from a source. This precautionary step was taken to ensure that any errors arising from incorrect component modeling were not carried back into the loop and propagated over time as the refrigerant repeatedly circulated through components. Once the open loop chiller was operating successfully and predictably, the circuit was then modified to create the normal closed-loop circuit. The simulation schematics and results in each instance are presented in the subsections that follow.

Parameter	Value
SHELL-AND-TUBE CONDENSER	
Sea water flow rate	0.0416 m^3/s
Tube-side average pressure	0.325 MPa
Inlet Water temperature	304.26 K
Outlet Water temperature	309.26 K
Condensation Pressure	1.021 MPa
Condensation Temperature	313.313 K
Outer Tube Diameter	0.01905 m
SHELL-AND-TUBE EVAPORATOR	
Fresh water flow rate	0.0454 m^3/s
Tube-side pressure	0.125 MPa
Inlet water temperature	283.54 K
Outlet water temperature	279.82 K
Evaporation Pressure	0.323 MPa
Evaporation Temperature	275.928 K
Outer tube diameter	0.01905 m
COMPRESSOR	
Design pressure difference	0.698 MPa
Synchronous Speed	3600 rpm

Table 2.1: York 200-ton chiller specifications

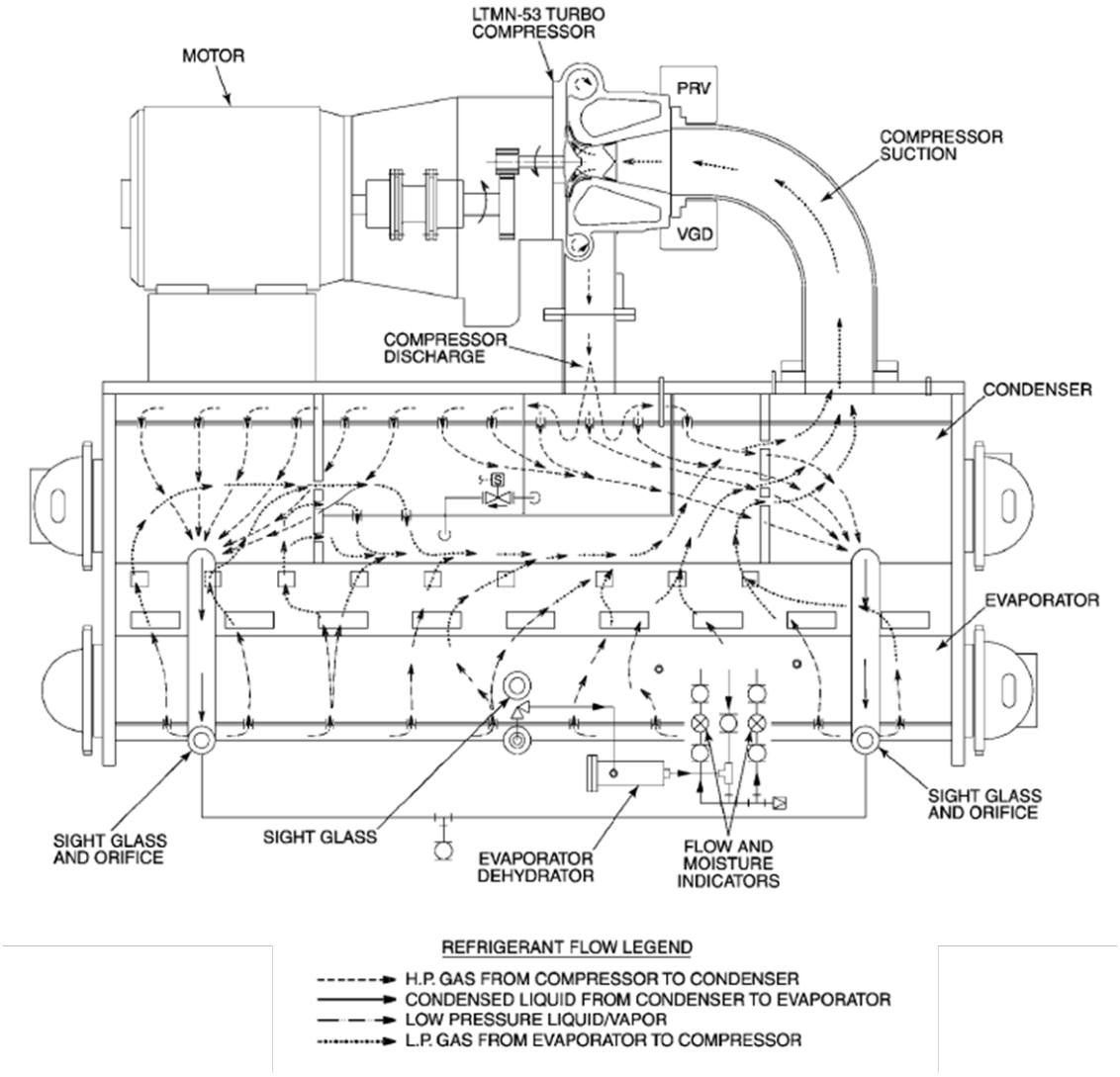

Figure 2.2: York 200-ton chiller with refrigerant flow diagram

2.3.1 Open-Loop Chiller

The open-loop chiller simulation allowed refrigerant to enter the chiller via the compressor inlet, at a user-defined state determined by the fluid pressure and enthalpy. The refrigerant then flows through succeeding components and exits into a fluid reservoir. Initially, the only component between the refrigerant source and sink was the centrifugal compressor. Thus the refrigerant outlet state from the model was capable of

being compared with a calculated exit state, based on York chiller parameters. When these results matched satisfactorily, the next component (the condenser and then the expansion valve) was successively added to the circuit. Once again, the refrigerant outlet state from the model was compared to the calculated state. In this way, successive components were added and tested to create a complete open-loop chiller model. This sequence of model enhancements is depicted using line diagrams in Figures 2.3a and b.

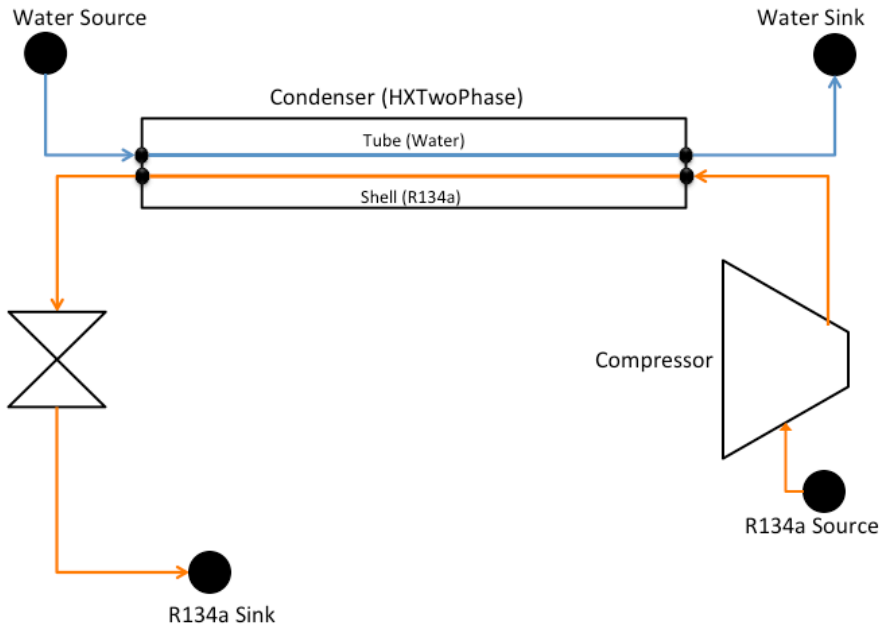

Figure 2.3a: Open-loop chiller test: compressor, condenser and expansion valve

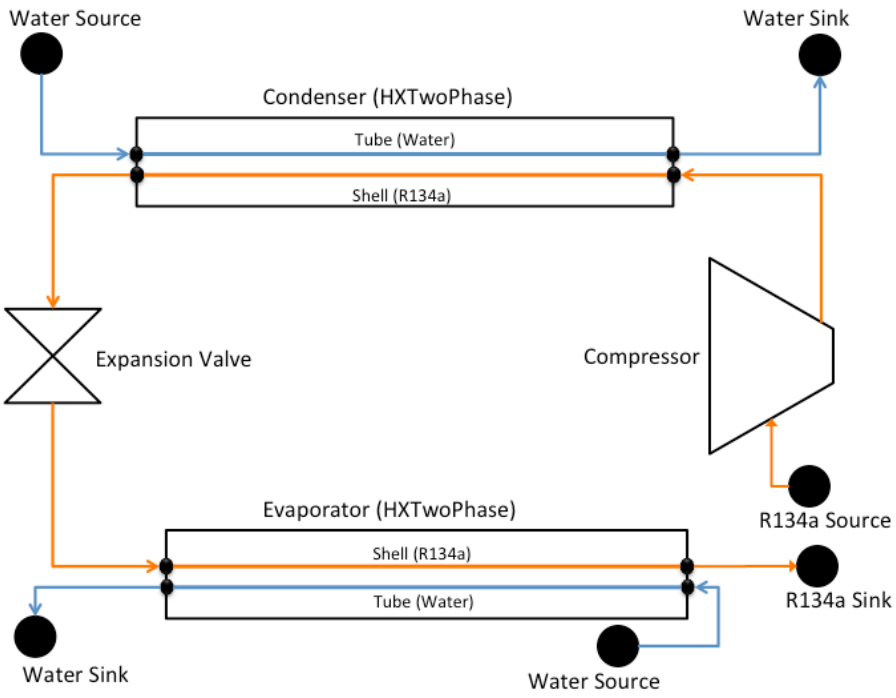

Figure 2.3b: Open-loop chiller test: complete open-loop simulation

Results produced during a typical, complete open-loop chiller simulation (as depicted in Figure 2.3b) are shown below. Simulation was carried out for a test case to 500 seconds, and the system was simulated from startup. Figures 2.4 and 2.5 below show the time evolution of the refrigerant enthalpy and pressure, at various points in the chiller. In each case, a steady-state is achieved rapidly. In Figure 2.4, the compressor and condenser exit enthalpies differ by the energy rejected in the condenser, the evaporator and TEV exit enthalpies differ by the energy added in the evaporator, and the compressor and evaporator exit enthalpies differ by the energy added by the compressor. Figure 2.5 shows the pressure differential across the condenser and evaporator. This figure also reflects the fact the evaporator exits to a reservoir at a fixed pressure.

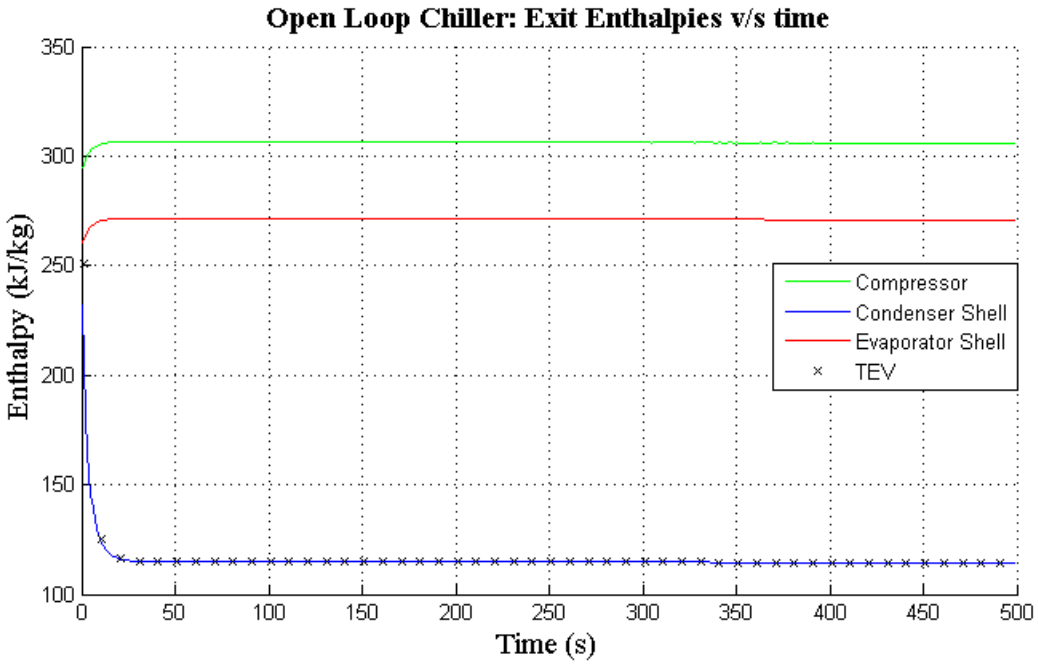

Figure 2.4: Exit enthalpies for the open-loop chiller simulation

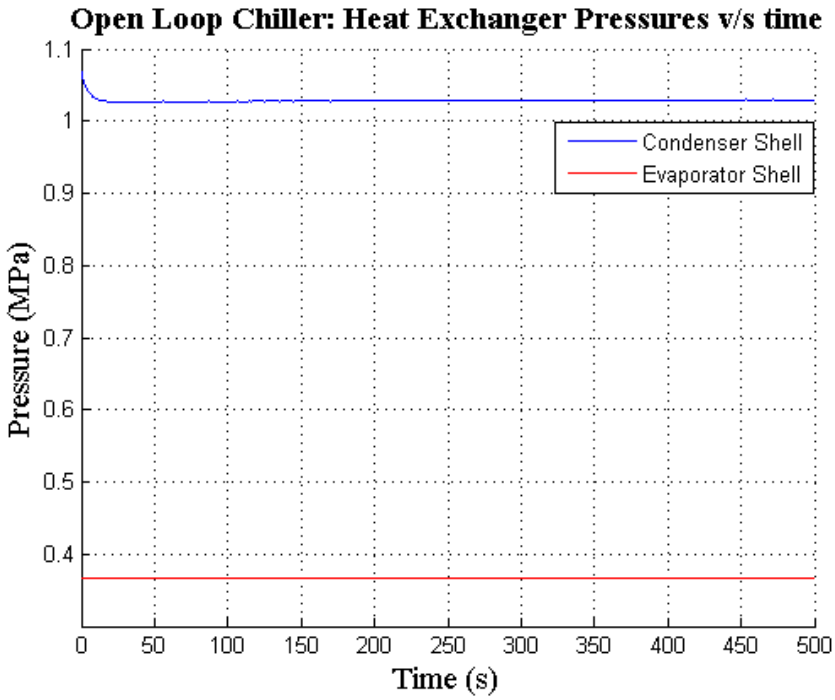

Figure 2.5: Exit pressures for the open-loop chiller simulation

Table 2.2 provides steady-state results for the refrigerant state at the exit of various components. As the model dictates, with no intermediate pipe losses, the exit enthalpy from the expansion valve is identical to the exit enthalpy from the condenser shell. Due to frictional losses in the heat exchangers, there is a noticeable pressure drop across the condenser and evaporator shell. The compressor receives refrigerant from the source at 0.323 MPa and pressurizes it to 1.03 MPa, resulting in a compression ratio of 3.19:1, very much in agreement with the actual compression ratio of 3.16:1. A steady-state condition is achieved within 50 seconds of startup in the open-loop case, which is relatively fast. This issue will be discussed further in the closed-loop chiller simulation.

Component	Pressure (MPa)	Enthalpy (kJ/kg)
Compressor	1.0296	305.9
Condenser Shell	1.0287	114.49
Expansion Valve	0.366	114.49
Evaporator	0.32346	270.97

Table 2.2: Open-loop chiller: steady-state component exit conditions

2.3.2 Closed-Loop Chiller Circuit

The open-loop simulations have methodically verified that component steady-state exit conditions agree with calculated conditions, and that the fluid reservoir condition comes to closure. Therefore, the refrigerant reservoir was eliminated and the evaporator exit was connected to the compressor inlet to close the circuit. A schematic of this configuration (with the hot gas bypass, heat load, and secondary fluid loops included)

is shown in Figure 2.6. This is the final chiller configuration. The hot gas bypass valve (HGBV) connects the condenser to the compressor to ensure a controlled level of superheat in the refrigerant entering the compressor.

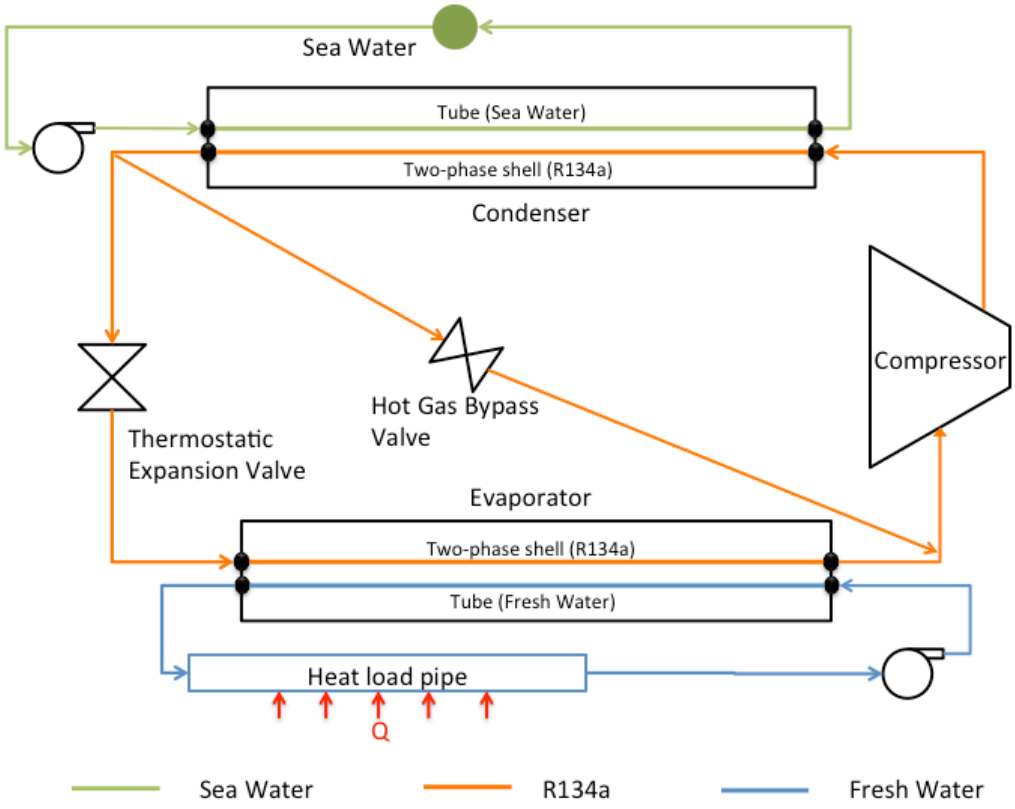

Figure 2.6: Schematic of a complete closed-loop chiller

In the absence of a source like that in the open-loop simulation, it is important to specify an initial fluid condition in the circuit as a reference for the refrigerant state at the beginning of a simulation. Also, it is necessary to choose one point in each loop as an independent pressure node, whose pressure can then be taken as a reference for the fluid solvers in calculating the pressures and flow rates throughout the loop. In the open-loop

simulations, this initial condition and the independent pressure node condition were both satisfied by the fluid reservoir.

Significant effort went into selecting an appropriate initial condition for the closed refrigerant loop. This condition is the initial fluid state at every point in the refrigerant loop; both a pressure and an enthalpy must be selected to uniquely identify this initial state. Common sense dictates, and numerical simulations confirm, that this initial state should be located on the low pressure side of the loop, i.e. at the evaporator. The fluid solvers in *DTMS* initialize a calculation by equating the inlet and exit condition for the evaporator. Suppose that the initial condition were chosen to be the equilibrium condition at the evaporator shell exit. This implies that there is a large deviation of fluid enthalpy between the initial condition and the equilibrium value for the shell inlet. This large deviation, when propagated in the closed loop, causes numerical difficulties and may give unrealistic simulation outputs. Similarly, if the equilibrium evaporator shell inlet were chosen as the initial condition for the circuit, then the large deviation between the initial condition and the equilibrium condition at the shell exit may give unrealistic simulation results. As a compromise, the initial starting condition was chosen to be that of saturated refrigerant vapor at the theoretical evaporation pressure. This condition is a pressure of 325 kPa and an enthalpy of 252.028 kJ/kg. To implement this initial condition, simulation logic is programmed at the evaporator exit node to output saturated vapor from the evaporator into the compressor for the first 100 simulation seconds, and act as a regular pressure junction thereafter. 100 seconds is sufficient time for the system to go from the

initial condition to approximately its equilibrium value, thereby eliminating numerical issues due to the large initial deviations described.

Selection of an independent pressure node in the refrigerant loop is also necessary. The pressure at this node determines the pressures and flow rate throughout the loop and cannot be altered by the solver as it is considered an input. However, Equation 2.33 shows that the evaporation pressure will change depending on the heat load in the secondary fresh water loop. Therefore, to establish an independent node for the solver and to allow that pressure to change based on the dynamic heat load, the evaporator inlet was assigned as the independent pressure node. Logic is programmed in the evaporator model to change this inlet pressure based on the dictates of Equation 2.33. For the fresh water loop, the initial fluid condition is the equilibrium evaporator tube inlet condition and the independent pressure node is the tube inlet pressure. Seawater in the seawater loop is considered to be a true reservoir and is modeled as an infinite source. Therefore, the initial condition and independent pressure node are both taken from this reservoir.

Figures 2.7 and 2.8 depict the time evolution of enthalpy and pressure at the exit of each component for a complete closed-loop simulation. In the results that follow, the simulation time is 1500 seconds and no controls are employed. The"*Eqm*" labels indicate equilibrium values at the settling time shown. For parameters that do not have a reference value (such as pressure and flow rate), a steady-state value with a maximum 1% fluctuation was considered as an equilibrium criterion. For temperature, the Kelvin scale was used as a reference and a fluctuation of 0.1 K about steady-state was considered as

an equilibrium criterion. For enthalpy, the ASHRAE reference scale was used and a fluctuation of 0.1 kJ/kg at steady-state was considered as an equilibrium criterion.

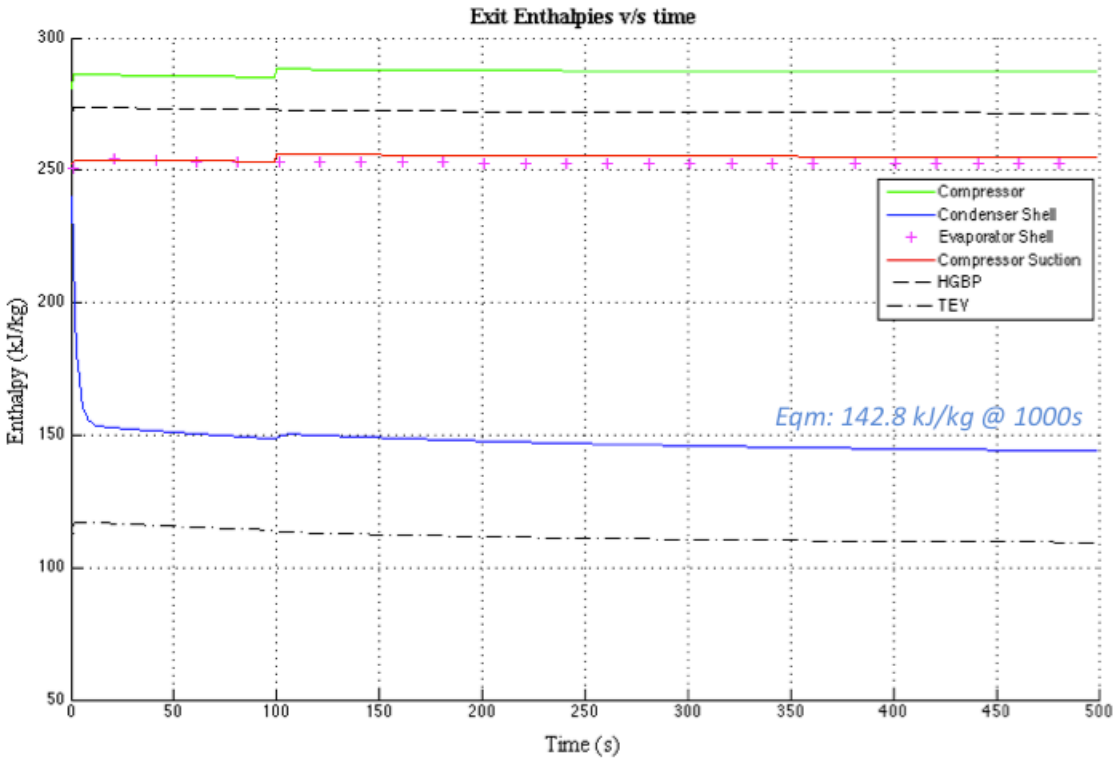

Figure 2.7: Exit enthalpies for the closed-loop chiller simulation

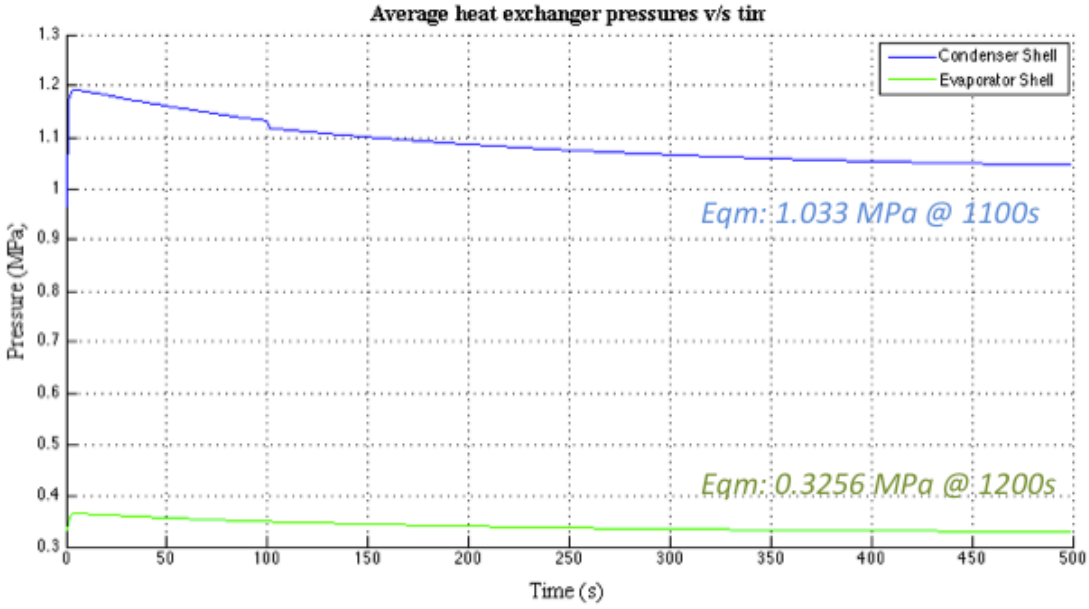

Figure 2.8: Exit pressures for the closed-loop chiller simulation

Based on these results, it is obvious in all cases that the variables plotted slowly approach a steady-state value, with some variables requiring more than a thousand seconds to achieve equilibrium. This time-delayed response is due to the heat-exchanger dynamics, and is an expected physical response of the system in the absence of controls. Specifically, the refrigerant exiting the evaporator shell enters the compressor, and any error between its current value and the steady value is propagated throughout the circuit. This error is slowly reduced over time until all components reach a steady-state. This was not the case in the open-loop simulation, since the fluid entering the compressor was provided from a reservoir, which was in a fixed state throughout the simulation. Therefore, the open-loop simulation was much faster in arriving at steady-state, about 50 seconds versus on the order of a thousand when compared to the closed-loop case.

At 100 seconds of simulation, there is a small, but noticeable and sudden, change in several of the plotted variables. This is induced by the evaporator exit node, which at 100 seconds converts to a pressure junction that is connected to, and takes fluid from, both the evaporator shell and the HGBP valve. At 100 seconds, this node does not enforce the presence of a saturated vapor at the evaporator exit and allows the HGBP valve to determine the superheat of the compressor suction fluid instead. The evaporator exit refrigerant is mixed with the HGBP valve exit refrigerant to produce the desired compressor inlet condition, which is refrigerant superheated by about 3-5 kJ/kg (3-5 K) above the saturated vapor state. This construct over the first 100 seconds allows each component to approach its steady-state condition, thus avoiding numerical issues associated with large initial deviations from equilibrium values.

Another important feature of this simulation is revealed in Figure 2.9, which plots only the first 25 seconds of this simulation. An initial jump in certain states during the first few seconds can be clearly seen. Specifically, exit enthalpy from the evaporator and condenser shell both begin at the uniform initial condition of 252.028 kJ/kg. Since the heat exchangers are modeled dynamically, the transition from this value toward their respective equilibrium values is gradual. However, the compressor and valves are modeled statically. Therefore, their enthalpies adjust to the inlet and outlet conditions nearly instantaneously. These initial fluctuations arise from a single starting condition that is applied throughout the refrigerant loop. It is a numerical issue that has been partially mitigated with the logic incorporated at the evaporator exit node.

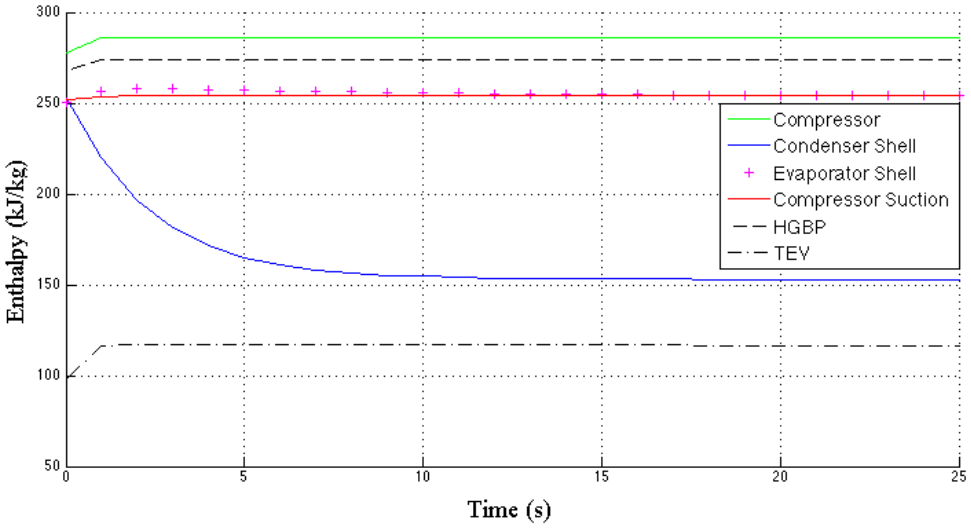

Figure 2.9: Exit enthalpies for closed-loop chiller simulation, first 25 seconds

Table 2.3 compares steady-state values with values that were obtained theoretically [8]. The steady-state results from the current *DTMS* simulation are in close agreement with the theoretical steady state values, varying by at most about 1%. The steady-state values for fresh water temperatures are within 0.12% of their theoretical steady-state values. The condensation and evaporation pressures are calculated using an average of the pressures at the respective heat exchanger's inlet and outlet. These are within a reasonable error bound, on the order of 1%, compared to the theoretical values.

Parameter (units)	***DTMS***	**Theoretical**	**% Diff.**
Evaporator refrigerant mass flow rate (kg/s)	4.95	4.897	1.082%
Condensing Pressure (MPa)	1.033	1.0210	1.175%
Evaporating Pressure (MPa)	0.3256	0.3230	0.8%
Fresh water outlet temperature (K)	279.5	279.82	-0.114%
Fresh water inlet temperature (K)	283.3	283.538	-0.084%

Table 2.3: Comparison of *DTMS* chiller simulation with theoretical steady-state values

Chapter 3: Functional Changes to the *DTMS* Framework

When the ESRDC was formed in 2002, the consortium immediately recognized the need for accurate modeling of various components of an All-Electric Ship and for simulations at the ship-system level. This philosophy clearly extended to the thermal management, where during extended deliberations numerous options were considered for dynamic modeling of thermal-mechanical-electrical systems and the various heat loads that they presented.

Commercial simulation software, such as *CycleTempo* [25] for steady-state conditions and *ProTRAX* [26] for dynamic conditions, was initially used for thermal management investigations. However, it was soon evident that the Navy peculiar subsystems under consideration would involve highly transient, dynamic loads and extreme conditions that could not be readily modeled using commercial software structured for an industrial environment. This prompted the thermal management team at the University of Texas at Austin (UT) to create an in-house thermal management tool, now known as Dynamic Thermal Modeling and Simulation (*DTMS*) framework, that provided both the developer and the end-user with complete flexibility and control over component and system-level modeling.

This framework was created by UT graduate student Patrick Paullus in 2007 [8] and has since been improved upon by several UT students, principally Michael Pierce, who revamped the initial framework to create an updated version of *DTMS* with significant additional features [24]. This chapter deals with functional improvements to

the *DTMS* framework that provide additional flexibility for the end-user while simultaneously addressing a needed upgrade of dynamic control aspects of the software.

3.1 Functional Changes for Debugging

The *DTMS* framework undergoes continuous improvement and debugging plays an important role in this process. Developers must be able to access the flow of data, in a way that is easy to follow through various aspects of a simulation, thus enabling them to analyze the behavior of additions to the framework and to track down errors or unexpected behaviors that are an inevitable part the development process. This section deals with improvements made to the debugging system in order to facilitate improved understanding of the flow of data during execution of a *DTMS* simulation.

In order to dynamically configure and execute complex physical simulations using a flexible input system, it is imperative that *DTMS* be able to extract meaningful data produced by various elements of the framework during runtime execution. Also in any programming language, and particularly for full object-oriented C++, it is important to understand how the various classes and their associated objects interact with one another during execution of a simulation. Therefore, the ability to "debug" user generated computer code is a fundamental tool for the software developer. Debugging tools are pieces of code that facilitate the process of monitoring the evolution of selected runtime parameters during software execution. These tools help the model developer to understand how intermediate values change during program execution and to look for errors in logic and/or process.

3.1.1 Terminal Outputs for Out-of-Bounds Variables

When certain repetitive calculations are performed, with one or more unworkable input variables, the results may easily go out-of-bounds. This means that something has gone wrong in the simulation and needs to be investigated and fully understood before proceeding further.

DTMS employs an established method to pause a simulation when variables of interest (as chosen by the user) go out-of-bounds [24]. The user is then issued a runtime warning that a model variable no longer has a finite value. This variable must then be traced within the debugging log file. At times, the log file can become very large with tens of thousands of lines of debugging text. Obviously, it is then difficult to find the variable at issue and trace the origin of the difficulty.

To improve upon this method of error identification, code has been added to identify a specific out-of-bounds variable at the application terminal. An example is shown in Figure 3.1 where the text output mentions both the symbol of the out-of-bounds variable (e.g., *P* for pressure) and the specific model in which the error occurred. It is then relatively simple for the user or developer to locate that variable in the debug log and trace its value back in time to pinpoint the cause of the error. If the log file is too large, then the developer may use existing functions within the *DTMS* debugging class to specify objects that should provide a debug output, obviously including objects already identified as out-of-bounds. Rerunning a simulation with these debugging specifications only changes the debug log and not the actual simulation, allowing the developer to go through a much smaller debug file to identify the source of a calculation error.

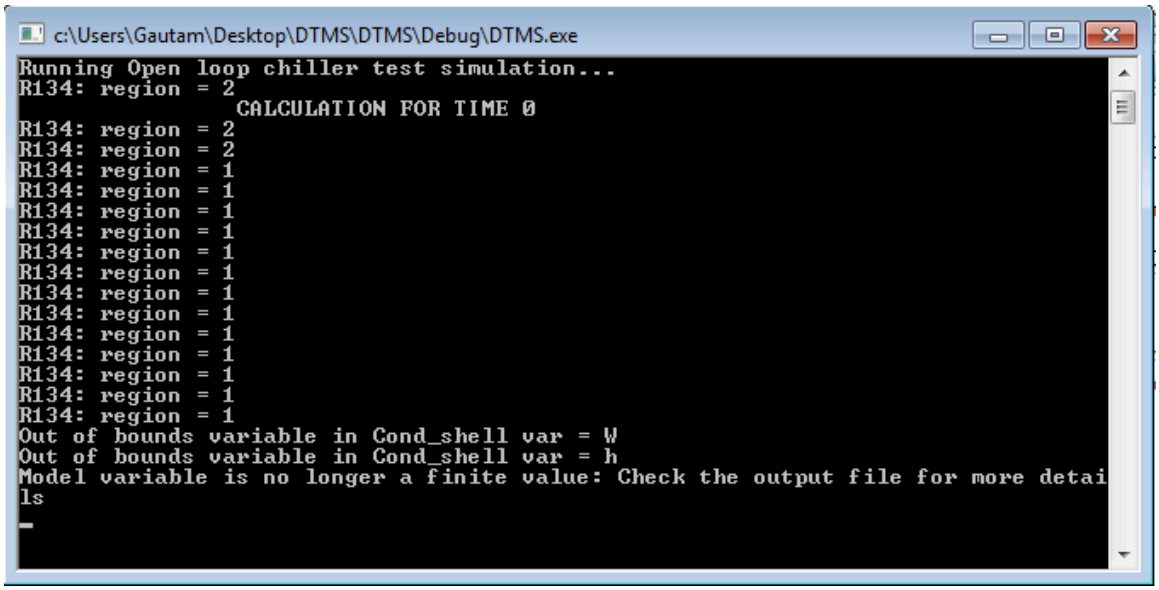

Figure 3.1: New terminal output for error logging

3.1.2 Terminal Outputs for Simulation Time, Fluid Properties, and Exit Codes

The previous *DTMS* debugging methodology focused primarily on the log file to trace any errors. However, there are times when the simulation itself may be very lengthy. Previously, *DTMS* provided no indication for the user or developer as to how long the simulation was taking, or whether it was "stuck" in a particular location during a simulation. If these errors occur in the starting few seconds of a simulation and are propagated over time, the final output file may contain a huge amount of information and give results that make no logical sense.

In an effort to make the flow of runtime information more transparent to the user, terminal outputs have been introduced to indicate the simulation time for the application. Also, every time the refrigerants properties are updated, the state of the fluid is output to the terminal in the form of a number (1 = saturated, 2 = superheated, 3 = subcooled). This technique gives the developer a rough idea of how far a simulation has progressed and

how the state of the refrigerant is behaving. A snapshot of the new terminal output is shown in Figure 3.2 below.

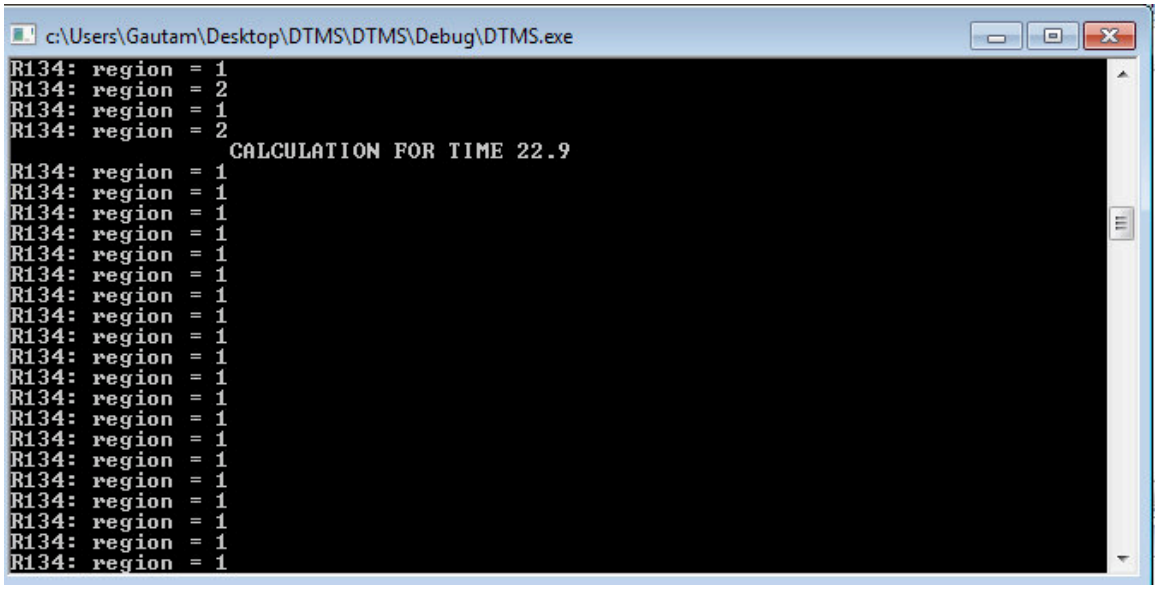

Figure 3.2: New terminal output during simulation runtime

If the fluid solver takes many tries to converge to a solution, then fluid properties will be updated many times before a final solution is achieved. This means that the refrigerant region is output to the terminal an unusually large number of times for such time steps.

Another added runtime feature is an output in seconds of the elapsed clock time for a simulation, if and only if a simulation is successful. This is expressed as an exit code via a return value of the `main()` simulation function. Generally, if the starting condition is near steady-state or reaches steady-state quickly, the Newton-Raphson fluid solvers will converge to a solution rapidly. The time taken to run a simulation is a soft indicator of how the simulation has progressed. The code to output the elapsed clock time must be added in every main simulation file coded by the user, and is explained in

Section 3.3. A screenshot of the result of a successful simulation is shown in Figure 3.3 where the elapsed clock time is 56 seconds.

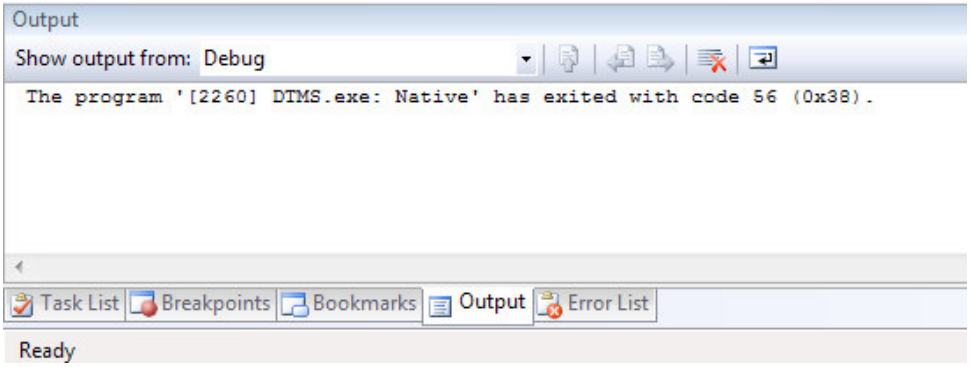

Figure 3.3: Exit code within the main program indicating clock time for simulation in seconds

A final important feature is the execution check for certain essential parameters. If there were an error for such a parameter, then the simulation would exit with a code of -1. This does not conflict with a positive exit code for a successful simulation. An example of this will be covered in the next section.

3.1.3 Units for Variables in the Debug Log File

The debug log file is a great tool created to enable the developer to track the flow of data after a simulation is completed [24]. The developer decides what simulation objects are output to the debug log or, more specifically, which functions will output variables and what variables are entered into the log. However, just the value of a variable itself may be misleading if the units of the variables are not specified. Thus, all newly created models now have their debug code modified to include the units and full name of the variable. Certain functions in older models have been updated to include

units in their debug output as well. Most calculations in *DTMS* are in metric units. When added to the already existing template of the debug file where the name of the model and the function in question is specified, this procedure removes confusion as to what variable is being considered, as well as its value and units. For the purposes of contrast an old and new debug log file are depicted in Figures 3.4 and 3.5. This inclusion of units is not a new or different type of code added to the framework, but merely implementation of a debugging practice. It would be strongly advised for future developers of *DTMS* to adhere to this practice to avoid confusion amongst variable meanings, values, and names.

```
[ThermalFluidEffortModel] pN10a: <calculateStates>   pressure_ = 125999
[ThermalFluidEffortModel] pN10a: <calculateStates>   enthalpy_ = 25.7788
[Water] : Entering updatePropsPH(double, double)
[Water] : <updatePropsPH> Inputs:
[Water] : <updatePropsPH>   press_ = 125999
[Water] : <updatePropsPH>   enth_ = 25.7788
[Water] : <updatePropsPH> Outputs:
[Water] : <updatePropsPH>   pressure = 0.125999
[Water] : <updatePropsPH>   temperature = 279.258
[Water] : <updatePropsPH>   enthalpy = 25.7788
[Water] : <updatePropsPH>   density = 999.952
[Water] : <updatePropsPH>   cp = 4.20239
[Water] : <updatePropsPH>   cv = 4.20176
[Water] : <updatePropsPH>   entropy = 0.0929656
[Water] : <updatePropsPH>   viscosity = 0.00146649
[Water] : Exiting updatePropsPH(double, double)
[ThermalFluidEffortModel] pN10a: <calculateStates> My Effort = 125999
[ThermalFluidEffortModel] pN10a: Exiting calculateStates()
```

Figure 3.4: Old debug log file, without units for variables

```
[R134] : Entering updateProps()
[R134] : <updateProps> Outputs:
[R134] : <updateProps>   pressure(MPa) = 0.32346
[R134] : <updateProps>   temperature(K) = 275.866
[R134] : <updateProps>   enthalpy(kJ/kg) = 252.028
[R134] : <updateProps>   density(kg/m3) = 15.8853
[R134] : <updateProps>   cp(kJ/kg-K) = 0.90988
[R134] : <updateProps>   cv(kJ/kg-K) = 0.768786
[R134] : <updateProps>   entropy(kJ/kg-K) = 0.93004
[R134] : <updateProps>   viscosity(Pa-s) = 1.08265e-005
[R134] : Exiting updateProps()
```

Figure 3.5: New debug log file, with variable names and units

3.2 FUNCTIONALITY IMPROVEMENTS IN *DTMS*

The previous section discussed improvements to the *DTMS* debugging process with the intent of assisting the developer in understanding the flow of data and the physical meaning of various parameters and their values. This section deals with changes in the *DTMS* framework that prevent potentially erroneous runtime execution.

3.2.1 Event Initiation in PID Control

The `setEvent()` function in the PID control model (class *CTLPIDController*) takes an input of a variable whose value must be modified and the time at which this event must occur. The previous implementation of `setEvent()` compares the values of the actual time and scheduled event time directly, i.e.,

```
if (scheduledEventTime = actualSimulationTime)
```

This coding can create problems when actual time values are stored as double-precision floating point values. For example, if `actualSimulationTime` is stored as 105 seconds and the `scheduledEventTime` is stored as 105.0000000001 seconds, then these two values will never match and the scheduled event will not occur. The simple code to address this issue is to check whether `scheduledEventTime` lies between `actualSimulationTime` and '`actualSimulationTime + timestep`', i.e.:

```
if (scheduledEventTime ≥ actualSimulationTime &&
scheduledEventTime ≤ actualSimulationTime + timestep)
```

It can be seen that in this case, that the use of double precision floating-point numbers would not create a problem, and the event will occur as scheduled. This is an example of a small and important change in *DTMS* to ensure proper event handling. The next two subsections discuss, in detail, two very important changes to the basic framework.

3.2.2 Fluid Property Updates Near the Saturation Line

This feature was added to address fringe effects in fluid property calculations, which is explained using the following example. Consider the fluid R134a condensing at a pressure *P*, and let it go into the subcooled region towards point A as shown in Figure 3.6 below. Suppose the temperature of the fluid at point A needs to be calculated, using the function `updatePropsPH()` which works using double interpolation.

The executable first determines the state of the fluid (subcooled, saturated, or superheated). In this case, the fluid is in the subcooled region. Then, the code accesses the fluid properties of subcooled R134 at discrete points stored in lookup tables within

the *DTMS* framework. These points are denoted as diamond points (◆) in Figure 3.6. The code compares and checks the range of pressure in which the point lies. In this case, P lies between P_1 and P_2, specific pressures at which fluid properties are stored. Next, the temperature at point A_1 is calculated between two ◆ points. Similarly, the temperature at point A_2 is calculated by using linear interpolation between two ◆ points. Finally, the temperature at point A is obtained via linear interpolation of the temperatures at points A_1 and A_2. Since all the required ◆ points are stored in the subcooled R134 fluid property tables, the calculation of temperature at point A is relatively straightforward.

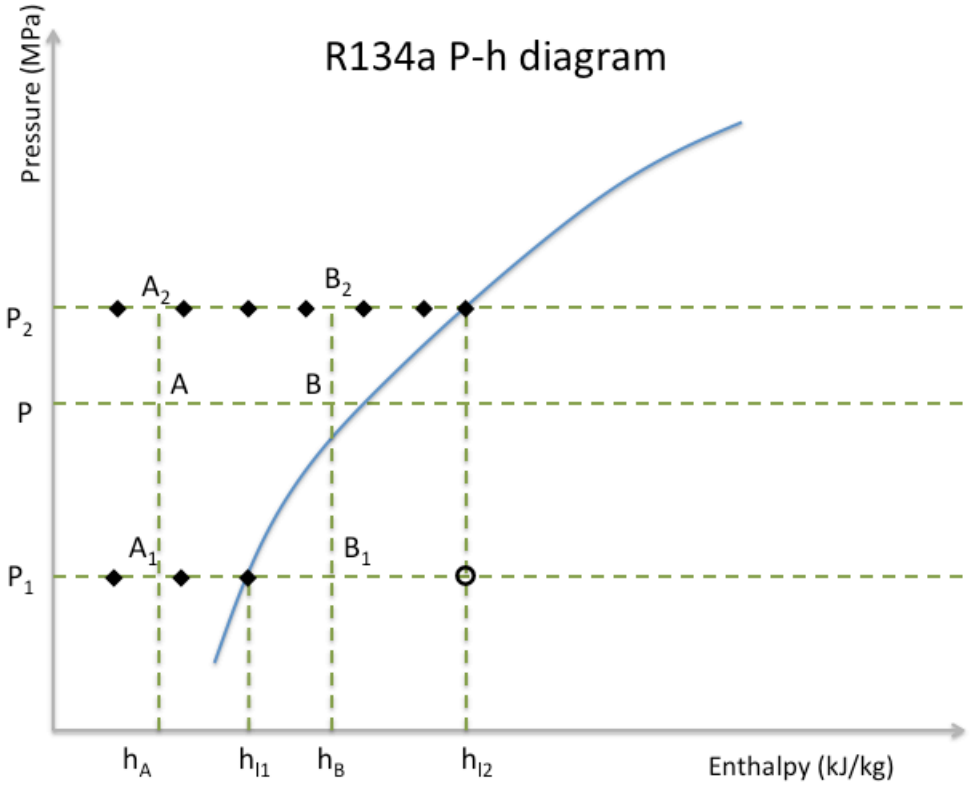

Figure 3.6: Fringe effect in fluid property calculations

Now consider a case where the fluid is at point B in Figure 3.6 and its temperature needs to be calculated using `updatePropsPH()`. Once again, the fluid state is in the subcooled liquid region and the code accesses the table of subcooled R134 property data. Next, it calculates temperatures at points B_1 and B_2 using linear interpolation between diamond points (◆) at their respective pressures. Calculation of temperature at B_2 is straightforward since there is a diamond point on either side. However, for point B_1 there is only a single diamond point on the left and none on the right. As a result, the executable interpolates using one diamond point and one erroneous value. This in turn gives an erroneous value for the temperature at B_1 and also at B. To avoid this problem when it occurs in a chiller simulation, an extra data point has been added to the property table of pressure P_1, shown as a circular point (O) in Figure 3.6. The fluid property values for this point are obtained through the saturated fluid property tables. As a result, point B_1 now has a sensible interpolated value for temperature, and also point B.

As can be seen from the example above, this erroneous output error only occurs for points like B that are very close to the saturation line. Thus, it is a fringe effect that has been addressed and corrected.

3.2.3 Simplification in Flow Rate Calculation

Occasionally, it may be necessary for the *DTMS* developer to specify a pre-determined rate of fluid flow through a particular circuit. This occurs when investigating parameters in another circuit connected to the given circuit. For example, in debugging of

a chiller model, secondary circuits (e.g., fresh water loop or sea water loops) may require fixed conditions in order to study fluctuations in the refrigerant circuit over time. In the current rendition of *DTMS,* heat input may be specified, but not flow rate or pressure since they are calculated using a Newton-Raphson solver.

In case of pressure, nodes may be defined as independent, meaning that their pressure is independent of fluid solver output and is instead taken as an input, or boundary condition, in determining the flow and pressure at other points in the circuit. In the case of flow rate, however, an entire circuit in series must have a common flow rate, and solving for this is performed by iteration in the solver. Since flow rate is not the output of a direct calculation, the exact flow rate is difficult to determine and the simulation must finish to obtain that value. There was no method to specify the flow rate for a particular model or circuit.

Therefore, a feature has been added to enable a device, such as the compressor, to have a pre-determined flow rate via a `setFlow()` function. All flow models that follow the square root flow-pressure relationship [8] of equation (3.1) may now be set with a pre-determined flow rate.

$$\boldsymbol{W = C\sqrt{\Delta P + s}} \tag{3.1}$$

Flow models used in the current chiller simulation, namely the centrifugal compressor, the expansion valve, the pipe, and the two-phase shell model, fall into this category. This feature will help future *DTMS* developers and users to test various models more efficiently, and also to facilitate the understanding of concepts other than the solver,

which is especially useful to developers in the initial stages of using *DTMS*. Notably, the `setFlow()` function must be applied in the main simulation file to all flow models in a series circuit. Once the user or developer has defined independent node pressures and exact flow rates, the solver will calculate intermediate pressures during a simulation, thereby making the simulation much faster.

3.3 IMPORTANT NOTES FOR THE *DTMS* END-USER

This section contains certain information that the end-user must keep in mind when running *DTMS* Simulations.

- As far as possible, when adding models to the *DTMS* Executive, you should add them in the order that they are arranged on the circuit. Thus, if a pipe comes at the end of a compressor, then the executive should first add the compressor model and then the pipe model immediately after.
- When modeling an expansion valve and setting valve position to an initial value for a simulation, the user must assign a fluid to the expansion valve first. This is necessary because when a valve position is calculated, the model also recalculates the flow coefficient to determine the fluid flow rate as follows

$$C_f = \frac{n_v}{n_{ve}} * W_{des} * \sqrt{\frac{\rho_{in}}{\rho_{in,des} * \Delta P_{max}}} \qquad (3.2)$$

This approach was explained in Section 2.2.2 where modeling the expansion valve was addressed. This function requires the density of the incoming fluid to

determine the flow coefficient. In the absence of a fluid assignment, the variable ρ_{in} has no assigned value and the executable returns an error.

- The simulation output is a comma separated value file (.csv), which is opened using Microsoft *Excel* by most users. If one wishes to view and store this data for generation of charts and other analysis, then the output file should be saved as a Microsoft *Excel* file (.xlsx).
- There is simulation code at the end of every simulation file in the `main()` function, which returns the real time required to run the simulation in seconds, to the closest integer. This should be present in the end-user's main simulation file. This code is given below and is also present in the *DTMS* tutorial.

```
        { …
        cBegin = clock() ;
        //Run the simulation executive
        Executive.runSimulation() ;
        cEnd = clock() ;
        cSim = (cEnd-cBegin) ;
        cSim = cSim/CLOCKS_PER_SEC ;
        Successful = true;}
    return int(cSim);
```

The last line (`return int(cSim);`) ensures that the return value is the simulation time, which has been changed from (`return 0;`) in the tutorial.

Chapter 4: Feedback Control

During dynamic simulation of integrated systems, it becomes necessary to introduce controllers to maintain the system in a desired state. For example, the inlet of a compressor should exhibit a controlled flow rate of vapor to prevent compressor surge – an unstable dynamic condition. Also, in the actual chiller, a controller should continuously monitor and control compressor suction. The nature and characteristics of these controllers are critical to successful simulation of dynamic performance. This chapter discusses PID (proportional-integral-derivative) control – a widely used, feedback control strategy for various industrial systems. The chapter is written in two parts, first presenting the fundamentals of PID controls and then the details of PID implementation in *DTMS*.

4.1 PID CONTROL

4.1.1 Fundamentals of PID

PID control is a common strategy used in industry to control various processes. A PID feedback control loop is based on controlling a single variable in the system to cause it to reach a pre-defined set point. It does so by subtracting the actual value of the variable from that of a set point to obtain an error. Based on the value of this error, adjustments are made to provide a control output. In order to make the set point comparison, the controlled variable must be something that can be readily measured or estimated. The control output is a variable that can be manipulated by the user in the

actual system. For example, one of the controls in a *DTMS* chiller simulation involves controlling the outlet temperature of the chilled water exiting an evaporator by manipulating the valve position of a thermostatic expansion valve. The basic logic for a PID control scheme is shown schematically in Figure 4.1 below.

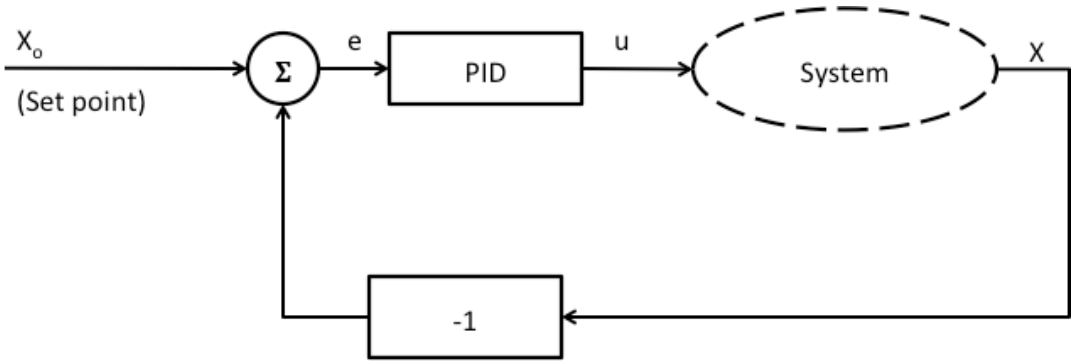

Figure 4.1: Block diagram for a feedback, PID control scheme

In the figure, X is the state variable of the system that is to be controlled. X_o is the set point defined for the variable X by the user. The summer adds the set point and the negative of the system state (-X) to calculate an instantaneous error.

$$\boldsymbol{e(t) = X_o - X(t)} \tag{4.1}$$

This error is then used as an input by the PID controller to provide the control output, u(t). It does so by using three constants, the proportional gain (k_p), the derivative time constant (τ_d), and the integral time constant (τ_i). The formula to convert the error into a control output is then [27]:

$$\boldsymbol{u(t) = k_p e(t) + k_d \frac{de(t)}{dt} + k_i \int e(t)\, dt} \tag{4.2}$$

where k_d and k_i are the derivative and integral gains respectively. These are related to the proportional gain and time constants by

$$k_i = \frac{k_p}{\tau_i} \tag{4.3}$$

$$k_d = k_p \tau_d \tag{4.4}$$

Thus, the control output computation reduces to

$$u(t) = k_p e(t) + k_p \tau_d \frac{de(t)}{dt} + \frac{k_p}{\tau_i} \int e(t)\, dt \tag{4.5}$$

4.1.2 Tuning of PID Variables

As can be seen from the equations above, the control output is dependent on three gains defined by the user before the simulation begins, namely k_p, k_d and k_i. Manipulation of these parameters determines the control output and the eventual response of the variable being controlled. This sub-section discusses each variable with illustrations to demonstrate the effect of adding each to the control algorithm.

Figure 4.2 shows a simple spring, mass, and damper system for which PID control is to be implemented [28]. The monitored variable is the displacement (x) of the mass (M) with the force (F) as the variable that is to be manipulated. In this example, the set point for x is 1 meter. The equation of motion for this mass is elementary and can be found in many dynamics textbooks.

$$F(t) = M\ddot{x} + b\dot{x} + kx \tag{4.6}$$

where the dots on x represent 1st and 2nd time derivatives, b is damping parameter, and k is a spring constant.

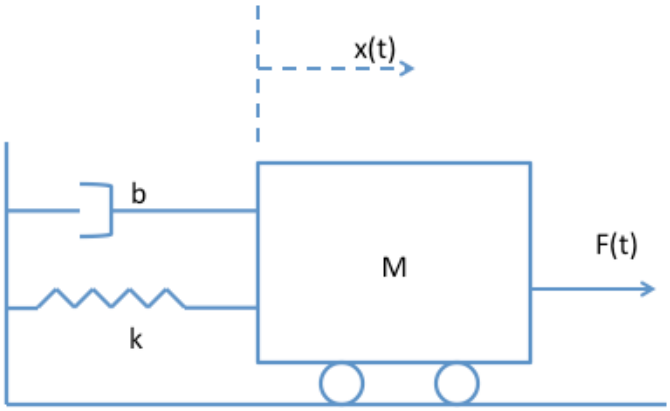

Figure 4.2: Mass, spring and damper system

For an open-loop system, i.e., a system controlled without position feedback, let $M = 2$ kg, $b = 10$ N-s/m, $k = 10$ N/m, and $F = 1$ N. The response of this open-loop system is seen in Figure. 4.3 below. The steady state position of the mass is $x_{ss} = 0.1$ m, i.e., a very large steady-state error of 0.9 m from the 1 meter set point.

Now, introduce a PID controller where the error in displacement is taken as an input and the applied force is adjusted accordingly. Let $k_p = 30$, $k_i = 0$ and $k_d = 0$, i.e., a control that is proportional only to the error. In this case, the controller calculates a closed-loop control output based purely on the error between the desired and actual displacement value. The system response is shown in Figure 4.4. The steady state error has now decreased considerably, and the settling time to reach steady state has been reduced to about 3 seconds, from about 5 seconds for the open-loop response. However, the closed-loop control has produced an overshoot above the steady-state value. Settling time, overshoot and steady-state error are common parameters used to characterize system response to a control algorithm.

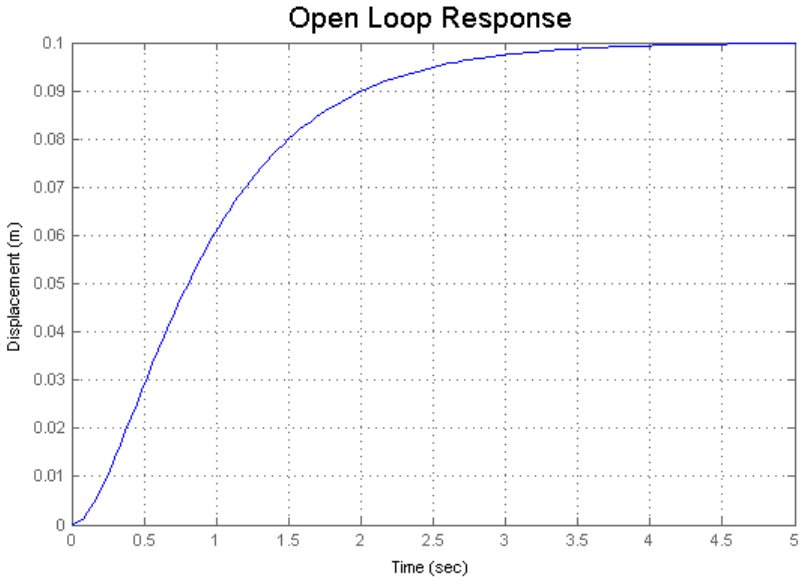

Figure 4.3: Open-loop response of a spring-mass-damper system

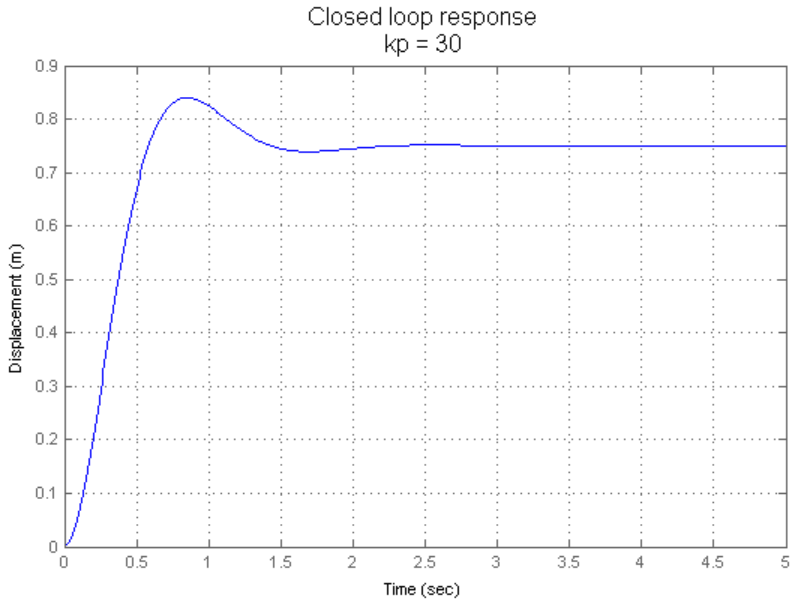

Figure 4.4: Closed-loop response for a spring-mass-damper system with $k_p = 30$

Now add derivative control, with k_p = 30 and k_d = 10, to make a proportional-derivative controller. The control algorithm calculates the output based on the error and

the first derivative with respect to time. The system response is shown in Figure 4.5. The overshoot is now absent and the settling time reduced to about 1.5 seconds. In general, adding derivative control reduces the settling time, steady-state error, and overshoot; although in this case the error is essentially unchanged. In essence, derivative control prevents very large values of *de/dt*. Thus, the slope of the response in Figure 4.5 is smoother than that in Figure 4.4.

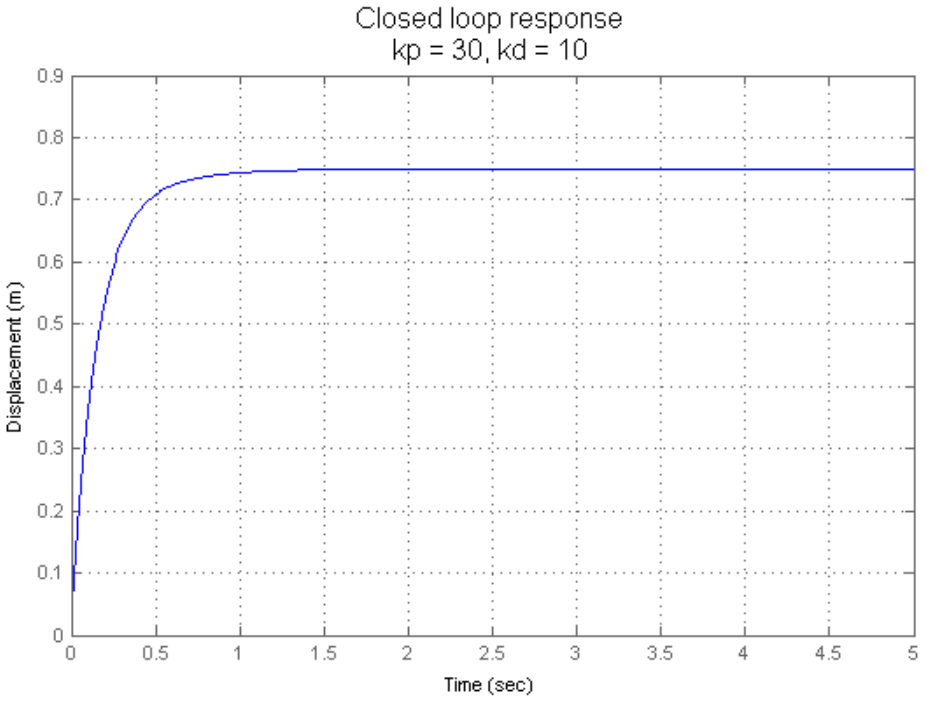

Figure 4.5: Closed-loop response for a spring-mass-damper system with k_p = 30, k_d =10

Consider now proportional-integral controller, with k_p = 30 and k_i = 30. This controller will operate on the error and it's integral over time. The system response is shown in Figure 4.6. The overshoot has now increased compared to the proportional case,

but the steady-state error has been completely eliminated. In general, an integral controller eliminates the steady-state error at the cost of reduced system response.

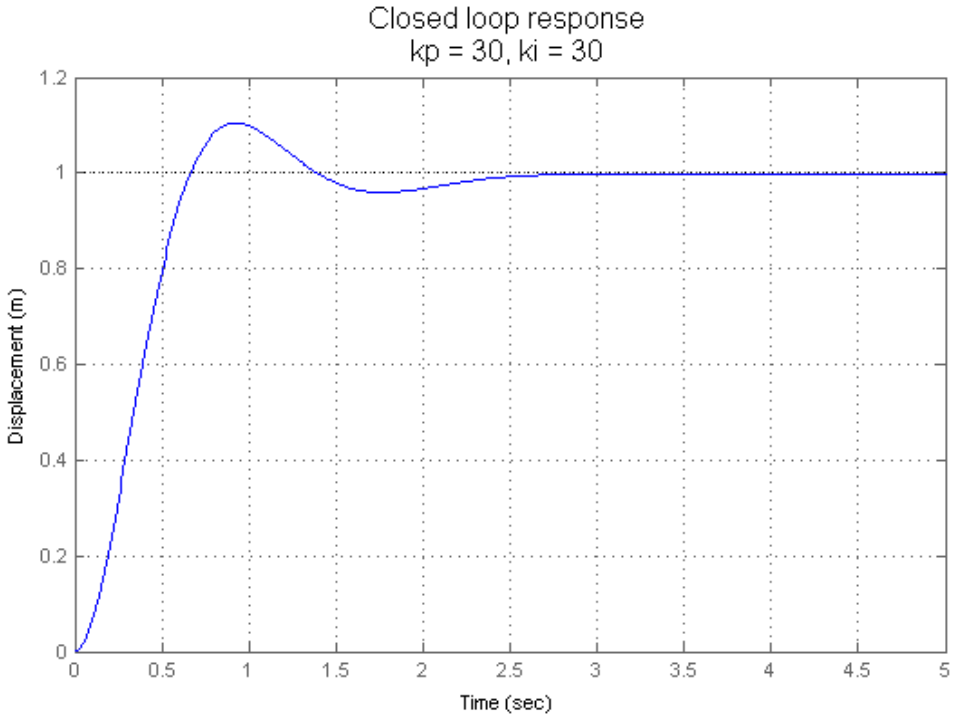

Figure 4.6: Closed-loop response for a spring-mass-damper system with $k_p = 30$, $k_i = 30$

Finally, consider now a full PID controller with $k_p = 30$, $k_i = 30$, and $k_d = 10$. Here, the overshoot aspect of the proportional-integral controller is compensated for by the derivative controller. The system response is shown in Figure 4.7. The maximum displacement is 1.006 m (a 0.6% error), and is reached within 2.5 seconds. Thus, even though the settling time to reach steady state has increased slightly, the steady state error has been completely eliminated due to the integral control. Table 4.1 shows the effects of

proportional, integral and derivative control on system response when properly implemented.

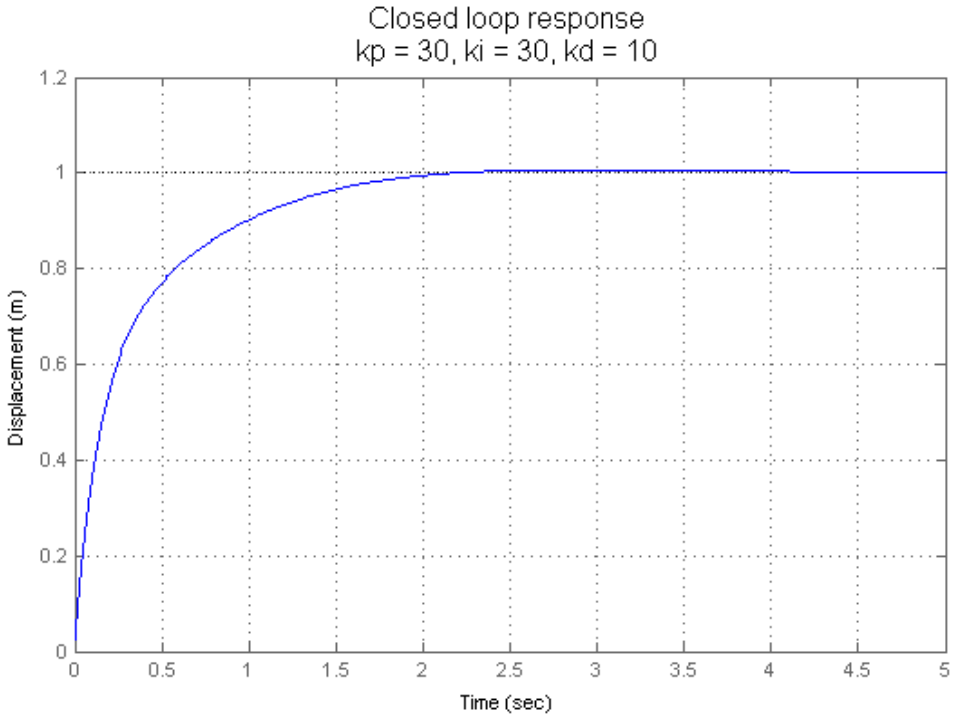

Figure 4.7: Closed-loop response for spring-mass-damper system with k_p=30, k_i=30, k_d=10

Controller	Overshoot	Settling Time	Steady-state Error
Proportional (k_p)	Increase	Small Change	Decrease
Integral (k_i)	Increase	Increase	Eliminate
Derivative (k_d)	Decrease	Decrease	Small Change

Table 4.1: Effects of proportional, integral and derivative control on system response

Clearly these controls are not completely independent of one another; changing one variable will change the effect of the other two. For example, reducing the proportional control may slow the system response and hence avoid large fluctuations of error with respect to time. In this case, the effect of derivative control, which works on

the slope of the error with time as an input, would be diminished. Thus, a table such as that above should only be used as a crude reference to both determine and assist in tuning the values of k_p, k_i, and k_d.

4.2 PID CONTROL IN *DTMS*

The previous section discussed the basic theory of PID control in order to consider how various aspects of this control approach affect the system response, and each other. This section deals with the implementation of a generalized PID controller in *DTMS* which is then tested on the closed-loop chiller model of Chapter 2.

4.2.1 *DTMS* PID Implementation Strategy

Implementation of PID (class *CTLPIDController*) in *DTMS* is based on a control class created for single-input, single-output systems, namely class *TransferFunction*, which is described briefly in this section.

TransferFunction is a child class of the generic class *DTMSControl*, which is the parent class of all control algorithms in *DTMS*. Class *TransferFunction* is based on the formulation of an algebraic transfer function between the system output and the input. The general transfer function between an input *x(t)* and an output *y(t)* is expressed as

$$\frac{\boldsymbol{Y}}{\boldsymbol{X}} = \frac{\boldsymbol{B_0 s^N + B_1 s^{N-1} + \cdots + B_N}}{\boldsymbol{A_0 s^D + A_1 s^{D-1} + \cdots + A_D}} \tag{4.7}$$

where ***Y(s)*** is the Laplace transform of the output and ***X(s)*** is the Laplace transform of the input. The variable ***s*** is the Laplace domain variable; it is the counterpart

of time in a time-domain representation. The numerator and denominator are generalized polynomial expressions of degree *N* and *D*, respectively.

The inverse Laplace transform of Equation 4.7 is written as

$$\boldsymbol{A_0}\frac{\boldsymbol{d^D y(t)}}{\boldsymbol{dt^D}} + \boldsymbol{A_1}\frac{\boldsymbol{d^{D-1} y(t)}}{\boldsymbol{dt^{D-1}}} + \cdots + \boldsymbol{A_N y(t)} = \boldsymbol{B_0}\frac{\boldsymbol{d^N x(t)}}{\boldsymbol{dt^N}} + \boldsymbol{B_1}\frac{\boldsymbol{d^{N-1} x(t)}}{\boldsymbol{dt^{N-1}}} + \cdots + \boldsymbol{B_N x(t)} \tag{4.8}$$

Notice that the i^{th} power of $\boldsymbol{s}$ in the Laplace domain becomes the i^{th} time derivative of the input and output variables in the time-domain. Although it may not always be possible to solve this differential equation analytically, it may always be approximated numerically in *DTMS* using a hybrid implicit-explicit Euler integration method [8].

In the case of PID control, equation 4.5 relates the input error *e(t)* and the control output *u(t)*. Taking the Laplace transform on both sides of this equation results in

$$\boldsymbol{U = k_p E + k_p \tau_d s E + \frac{k_p}{\tau_i}\frac{E}{s}} \tag{4.9}$$

where $\boldsymbol{U}$ and $\boldsymbol{E}$ are the Laplace transformations of *u(t)* and *e(t)* respectively. Thus,

$$\boldsymbol{\frac{U}{E} = k_p + k_p \tau_d s + \frac{k_p}{\tau_i s}} \tag{4.10}$$

$$\boldsymbol{\frac{U}{E} = \frac{k_p \tau_d \tau_i s^2 + k_p \tau_i s + k_p}{s\tau_i + 0}} \tag{4.11}$$

where the **0** in the denominator is added to indicate that the constant coefficient, $\boldsymbol{A_D}$ from equation 4.7, is 0.

The PID controller in *DTMS* is modeled as a special case of the transfer function control scheme. Two functions, `setDevice()` and `setMeter()`, that are specific to the control scheme in class *CTLPIDController* provide the means to specify the controlled and monitored variables, respectively. The output for the controlled variable may occasionally take on values that are not physically meaningful or out-of-bounds for the device in question. For example, the valve position of an expansion valve can only be set from 0 (fully closed) to 100 (fully open). Thus, the functions `setCeiling()` and `setFloor()` are used to establish upper and lower bounds for the control output. The functions `setDependentModel()` and `setOnOffControl()` are also provided to allow multiple control loops to work simultaneously. Finally, there are a variety of other self-explanatory functions, such as `setSetpoint()`, that provide the user with flexibility in defining control parameters. These functions are all described in [9].

Two other functions in this and every control class are significant. The `initialize()` function, located within the class itself, is similar to the `initialize()` function of other *DTMS* models in that it allows for the calculation of various internal parameters upon simulation startup. In the case of PID control, this function establishes the variables and arrays necessary for class *TransferFunction* to integrate Equation 4.8 numerically. It also sets the device initial condition specified by the user. The `simControl()` function is the main output calculation function; it is called at each time step to calculate output feedback based on the error obtained. Again, these functions are all described in [24].

The function inputs and calculation procedures in the *DTMS* control class do not require any fluid identification, i.e., the procedure only requires particular variables from particular models and not any fluid properties. Also, there are no directional assignments necessary, i.e., there is no reference value to any of the input or output models. Connected models are simply referred to as the "device" and the "metered" model. In the absence of fluid and directional assignments, the control system is typically coded just before the models are added to the resistive network solvers and simulation executive.

4.2.2 PID Implementation on a Marine Chiller

The PID control architecture in *DTMS* was tested using the chiller simulation presented in Chapter 2. In a marine chiller where the heat loads in the fresh water loop may vary considerably, the resultant energy exchange in the evaporator also varies considerably. To prevent too much or too little refrigerant cooling in the condenser when compared to the heat gain in the evaporator, it is often necessary to control the heat exchange in the condenser. This control is intended to ensure the presence of some amount of saturated vapor refrigerant with which the hot gas bypass valve may operate, while also maintaining a sufficiently low refrigerant vapor content exiting the condenser. Therefore, a PID control is often used to monitor the exit enthalpy of the condenser refrigerant and control heat exchange in the seawater loop. The monitored variable is the exit enthalpy from the condenser shell. The manipulated variable, or controlled variable, is the speed of the condenser pump. This variable speed pump controls the flow rate of water in the seawater loop, thereby controlling heat exchange between the two fluids.

The code used to initialize the controller and specify its parameters is shown in Figure 4.8. The `setMeter()` function is used to specify the monitored variable, by identifying the model and providing the specific controlled parameter as arguments. The set point for the enthalpy is 110 kJ/kg, corresponding to a quality of 0.014 (1.4%) at the *DTMS* calculated condenser pressure of 1.016MPa. The `setCeiling()` and `setFloor()` functions determine the maximum and minimum desired rpm of the pump. No data for the speed range of the pump was available; therefore, these limits were arbitrarily selected around the design speed of 3600 rpm. The specific control parameters (in this case: k_p = 6, τ_d = 1, and τ_i = 20) were selected by referring to Table 4.1 and attempting to ensure the best system response, without overshoot, subject to the physical constraints of the pump, e.g., speed range and reduced fluctuation of pump speed.

```
//Create PID Control for Cond Shell
CTLPIDController coolantControl(6, 1, 20); //Arguments:(kp, td, ti)
coolantControl.setName("CondEnthalpyControl");
coolantControl.setMeter(Condenser.getShellModel(), ENTHALPY);
coolantControl.setSetpoint(110); //kJ/kg
coolantControl.setDevice(CondPump, SPEED); //Manipulate speed of condenser pump
coolantControl.setCeiling(4000); //rpm
coolantControl.setFloor(3000); //rpm
coolantControl.setDeviceInitialCondition(3600);//rpm //Default initial condition
```

Figure 4.8: Code to create a PID controller for control of condenser exit enthalpy

The system response to this control scheme and the control output are shown in Figure 4.9. The data point shown marks the first instant, 579 seconds, at which the enthalpy achieves the set point, 110 kJ/kg. Figure 4.10 shows the control output variation over time. Smooth variation of the pump speed to reach the equilibrium value of 3700 rpm is apparent.

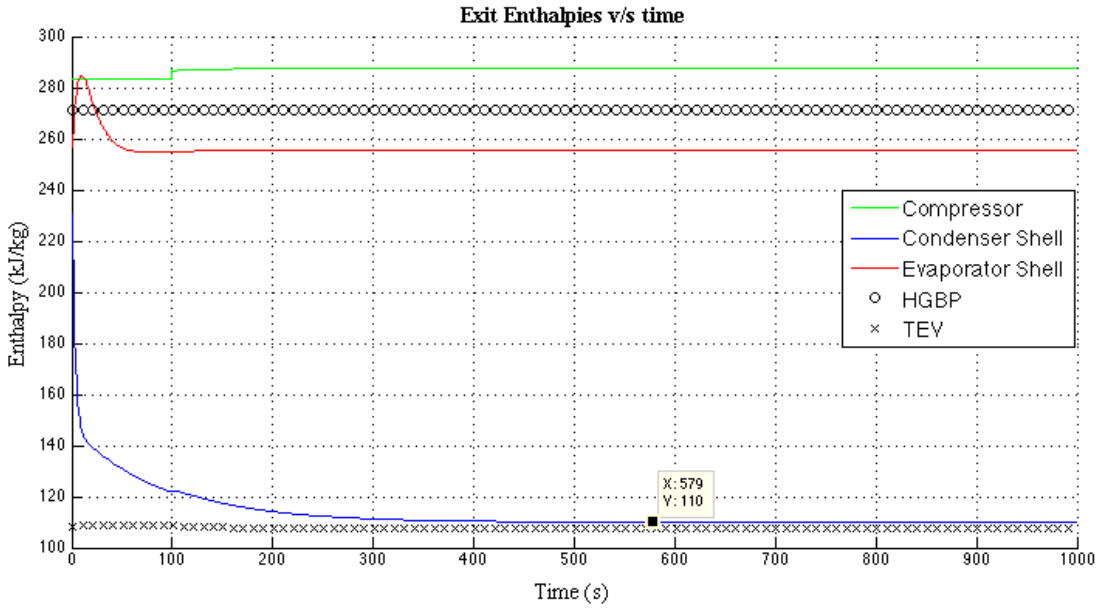

Figure 4.9: Exit enthalpies of components vs. time for PID control of condenser shell

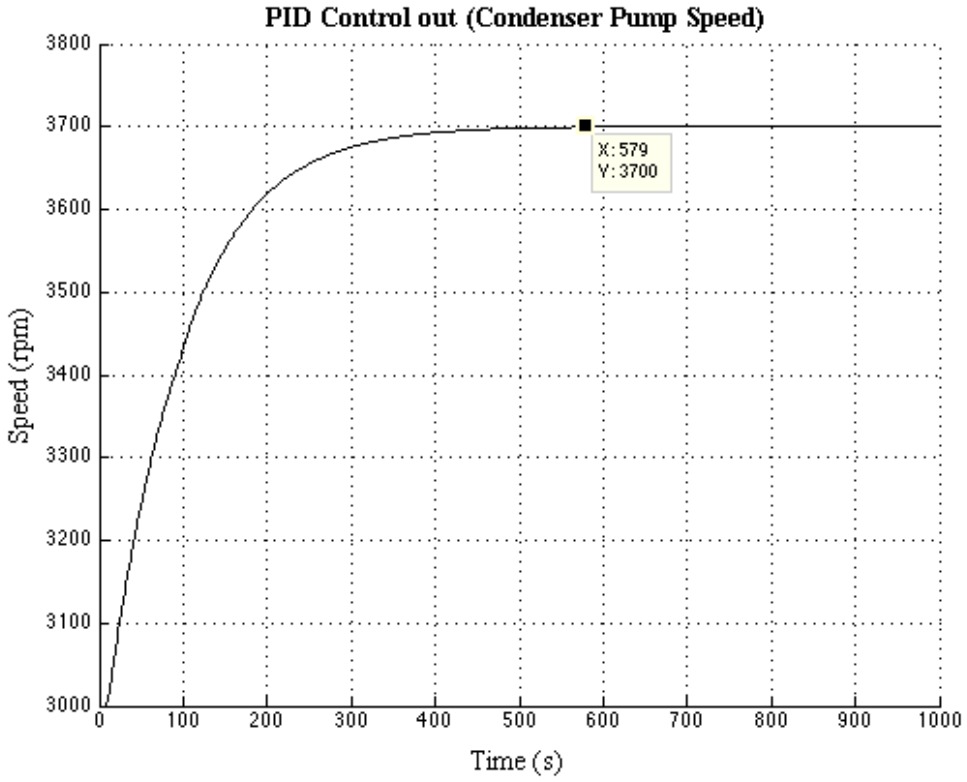

Figure 4.10: Condenser speed (control output) vs. time

This chapter has introduced the fundamentals of PID control, its necessity, and its implementation in a marine chiller model in *DTMS*. With this, the discussion of PID feedback loops concludes and the next chapter moves on to consideration and implementation of modern control theory in a marine chiller.

Chapter 5: Model-Based Controller Design

Previous chapters have focused on system-level modeling and simulation of a York chiller with a 200-ton refrigeration capacity. Rudimentary controls were then implemented using a PID feedback loop. During this development, it was noted that PID feedback is by nature local since it depends on the error between a given variable and its set point, regardless of the actual system dynamics.

Achieving the objective of a single PID loop may create an undesirable effect in another part of the system, while multiple PID loops may well conflict. This may lead to system instabilities that prevent the system from reaching a true steady state. Previous methodologies in *DTMS* included creating primary and secondary PID loops [8]. In this situation, there is a primary loop that is active while the secondary loop is inactive. When the primary loop has maintained its monitored variable within the tolerance limit of its set point for a few time steps, the two controllers switch positions and the new primary controller becomes active. In this "leapfrog" manner of control, two variables may fluctuate about their respective set points. The introduction of additional PID control loops would complicate this behavior and potentially render this "leapfrog" methodology totally ineffective.

To deal with a complicated component such as a chiller with its multiple controls, it was found necessary to do away with the strategy of implementing localized PID loops and explore other options. A heuristic control strategy was considered, where a controller would take state information from various locations in the chiller and compute a control

output accordingly. Although this worked for the current chiller simulation, the implementation of this method would vary from system to system. There is no general algorithm to access system variables and produce control outputs. Also, this strategy was implemented in conjunction with PID control, which reintroduces all of the issues, and disadvantages, associated with single or multiple PID loops.

Based on the above understanding of the problem, it was decided that full state feedback would be necessary to design a controller for the chiller system. This controller would be able to "understand" the interactions between various elements of the system and produce control outputs while considering global system response. Manual pole placement was considered, where a linearized system model is created about an equilibrium point. The eigenvalues (or poles) of the closed-loop system are placed according to the desired system response and open-loop eigenvalues [29]. These poles then determine the feedback control values for the system inputs. However, determining these gains heuristically, especially without specific response criteria for overshoot, settling time, etc. would be problematic. Therefore, it was decided to use a linear quadratic regulator (LQR) formulation from optimal control theory. This approach sidesteps manual pole placement in favor of weighting of system states and inputs. Thus, it provides a systematic way of influencing how gains are determined.

In the LQR formulation, a system-level control model computes outputs in a manner that takes into account contrasting effects of various outputs. Furthermore, the process for creating a linearized model, as well as its use to design a controller and determine control outputs, is extendable to larger or smaller systems. The details of this

theory are provided in various textbooks on the subject, most notably [30 and 31]. These references have been used throughout this chapter.

This chapter discusses the procedure for designing a controller for a large system, using the York 200-ton chiller as the system being controlled. It describes the process of identifying the minimal number of states required to describe the system, applying conditions to generate a linear system model for small deviations about equilibrium, and obtaining control outputs based on this linearized model. Results of this control strategy are shown and compared with chiller simulation results from both no-control and PID control cases. This chapter ends by addressing some limitations of optimal control theory and cases where it is not advisable to use this formulation.

5.1 SYSTEM MODEL LINEARIZATION

Linearization of a system model is a pre-requisite for applying optimal control theory to any system, linear or non-linear. The process involves the following steps to create the simplest linear model that can describe the entire system:

1. Determine the minimum number of critical states and control inputs that describe the entire system. These states are called “system states”.
2. Determine expressions that define the rate of change of the system states in terms of other states and control inputs.
3. Determine the equilibrium point for the model and linearize the state equations about this point to obtain a linearized system model for deviations about the equilibrium state.

5.1.1 Determining System States

This sub-section addresses the minimal number of critical states and control inputs that describe the entire chiller. For clarity, a chiller schematic is again shown in Figure 5.1 below. For reference, inlet and outlet states have been labeled for each device in the system. To determine the minimum number of critical states for the chiller, the interaction of various parameters at each of the 12 state points was studied to establish which variables change dynamically with respect to each other and which variables are static, time-invariant functions of others.

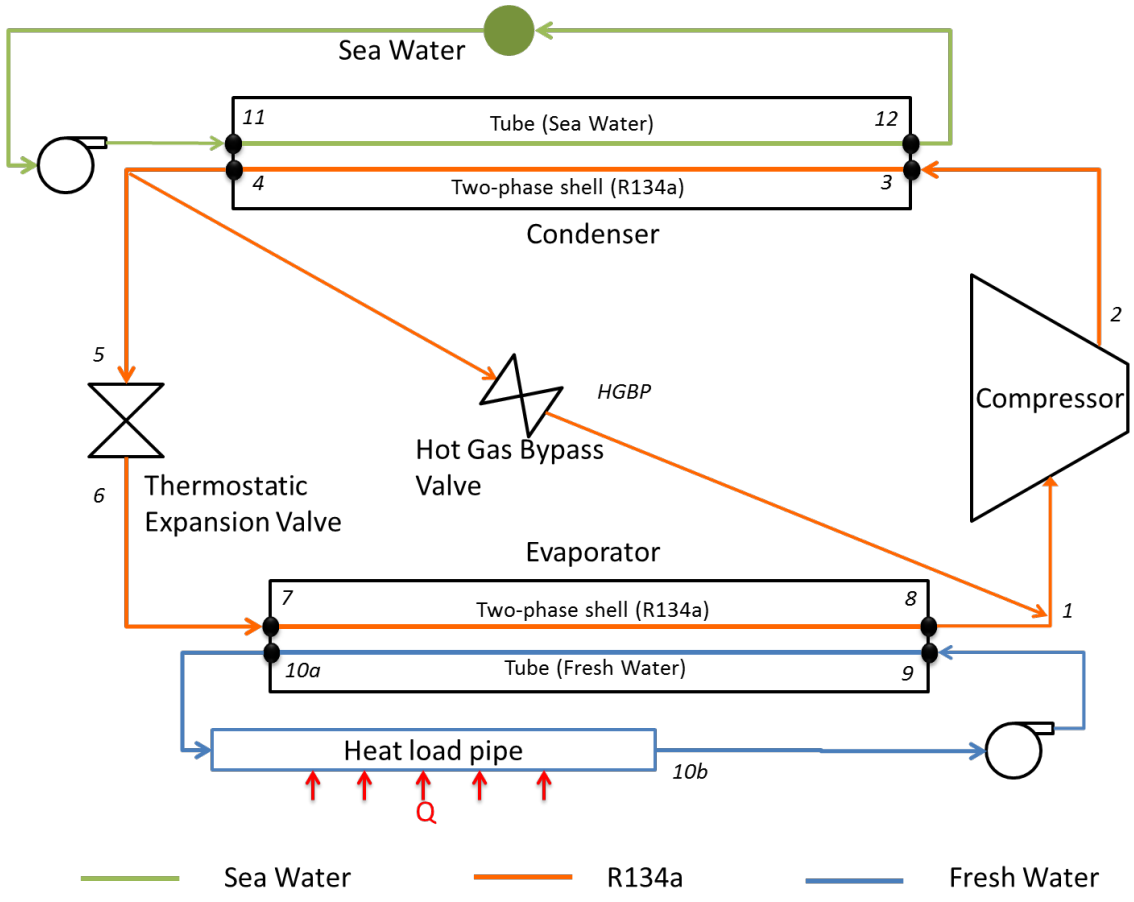

Figure 5.1: Schematic of chiller

To describe the chilled fresh water loop, the temperature of the water exiting the evaporator (T_{10a}) is considered a system state as its deviation about the equilibrium set point of 279.82 K (or 44 ^{o}F) is of utmost importance to the control scheme. Fresh water input pressure is an independent node, and that pressure (P_9) is fixed at 0.126 MPa, slightly above the average evaporator tube-side pressure of 0.125 MPa. The fluid condition at any point in the chilled water loop can be described in terms of P_9 and T_{10a}. The tube-side inlet temperature (T_9) is a static function of T_{10a} and the heat load (HL) is obtained by using a basic energy balance:

$$\boldsymbol{T_9 = T_{10a} + \frac{HL}{(WC_p)_{fw}}} \tag{5.1}$$

Here, $(WC_p)_{fw}$ is the heat capacity of the fresh water, i.e., the product of the mass flow rate and specific heat capacity of the liquid water.

In the seawater loop, the pump controls the flow rate. The pump model in *DTMS* was established in [8] and its flow rate ($W_{condpump}$) is a function of the pump speed (ω)

$$\boldsymbol{W_{condpump} = W_{des}\sqrt{\frac{\rho_{in}}{\rho_{des}(\Delta P_{max} - \Delta P_{des})}}\sqrt{\Delta P + \Delta P_{max}\frac{\omega^2}{\omega_{des}^2}\frac{\rho_{in}}{\rho_{des}}}} \tag{5.2}$$

where the subscript *des* corresponds to the design condition and the subscript *in* corresponds to the inlet condition. ΔP_{max} is the maximum possible pressure differential across the seawater pump.

In the refrigerant loop, two components are modeled dynamically: the condenser and the evaporator. Therefore, the states that describe the refrigerant at the inlet and exit of these heat exchangers (i.e., P_3, h_3, P_4, h_4, P_7, h_7, P_8 and h_8) are of specific interest.

Varying heat load conditions change the refrigerant evaporation and condensation pressure, and therefore the pressure range in the compressor, valves, and piping. Changing these, and the flow rate across any of the valves, changes the outputs of the Newton-Raphson solver used to calculate the flows and pressures throughout the circuit.

In the absence of exact formulae to determine these pressures and flows, one resorts to "system identification". This is a formal process used in control engineering that examines the modeled system equations to establish various inter-dependencies. For example, based on results from many simulations, it is observed that the pressure at pressure node 1 (P_1) is nearly equal to the pressure at the evaporator shell exit (P_8). Also, the compressor is modeled statically and hence its exit pressure (P_3) is a static, time-invariant function of the inlet pressure (P_1). Thus, P_3 is essentially a static function of P_8.

$$\boldsymbol{P_3 = f(P_8)} \tag{5.3}$$

Similarly, the pressure drop across the heat exchangers for various chiller conditions is a static function of the pressures themselves. Thus:

$$\boldsymbol{P_7 - P_8 = f(P_7)} \tag{5.4}$$

and

$$\boldsymbol{P_8 = f(P_7)} \tag{5.5}$$

Also,

$$\boldsymbol{P_3 - P_4 = f(P_3) = f\big(g(P_8)\big) = f\Big(g\big(h(P_7)\big)\Big)} \tag{5.6}$$

and

$$\boldsymbol{P_4 = f(P_3) = f\big(g(P_8)\big) = f\Big(g\big(h(P_7)\big)\Big)} \tag{5.7}$$

It is then clear that the evaporation inlet pressure (P_7) is a system state since it establishes the pressure at all other points in the refrigerant loop as static function of itself.

The enthalpy at various points in the refrigerant loop may be also related in a static sense. The fluid state at the entry to the thermostatic expansion valve (TEV) and the hot gas bypass valve (HGBP) are a saturated liquid and saturated vapor, respectively. The expansion valves are modeled as being constant enthalpy, i.e.:

$$\boldsymbol{h_7 = h_{TEVin} = h_l(P_4) = h_l\big(f(P_7)\big)} \tag{5.8}$$

and

$$\boldsymbol{h_{HGBP} = h_g(P_4) = h_g\big(f(P_3)\big)} \tag{5.9}$$

The mass flow rate for these valves is a function of the valve position. System identification reveals that the flow rate across each valve is a function of both the TEV and the HGBP valve positions (n). Then:

$$\boldsymbol{W_{TEV} = f(n_{TEV}, n_{HGBP})} \tag{5.10}$$

$$\boldsymbol{W_{HGBP} = f(n_{TEV}, n_{HGBP})} \tag{5.11}$$

The flow to the compressor is simply the sum of the flows across these valves:

$$\boldsymbol{W_{Comp} = W_{TEV} + W_{HGBP} = f(n_{TEV}, n_{HGBP})} \tag{5.12}$$

These enthalpies and flow rates may be used to calculate the enthalpy at pressure node 1, the compressor inlet, as follows:

$$\boldsymbol{h_1 = \frac{W_{TEV} * h_8 + W_{HGBP} * h_{HGBP}}{W_{TEV} + W_{HGBP}} = f(n_{TEV}, n_{HGBP}, h_8)} \quad (5.13)$$

Since the compressor is modeled statically, its exit enthalpy is a static function of the inlet enthalpy (h_1) and pressure (P_1). Now, h_1 is the enthalpy of the compressor suction, which needs to be maintained at about 3-5 kJ/kg (about 3-5 K) of superheat to prevent liquid refrigerant from entering the compressor. Thus, h_1 stays nearly constant, and h_3 varies predominantly with P_1. As established earlier, P_1 is nearly equal to P_8, and from Equation 5.5, P_8 is a function only of P_7. Therefore, h_3 can be shown to be a function of the evaporation pressure (P_7), i.e.:

$$\boldsymbol{h_3 = f(P_7)} \quad (5.14)$$

The four critical system states (P_7, h_4, h_8 and T_{10a}) are now established; they can be used to describe all other properties in the chiller system. Note that this selection of system states is not unique. For example, h_8 could be expressed statically in terms of h_1 using Equation 5.13. Then, h_8 and all system variables that are functions of h_8 could be expressed as functions of h_1. This would be further supported by the fact that h_1 needs to be maintained in a slightly superheated state for the sake of compressor suction. Then the control formulation would have a fixed set point 3-5 kJ/kg greater than the saturated vapor enthalpy state. The next section will establish that the current selection of states provides the simplest expressions for derivatives of the system states.

5.1.2 Determining State Equations

Now that the system states and control inputs have been established, the next step in creating a linearized system model is to develop equations for the system state derivatives in terms of the system states and other known quantities.

Since the heat exchangers are modeled dynamically, their exit enthalpies are dependent on system variables and also on time. From Equation 2.15 for the exit enthalpy in a two-phase evaporator shell model:

$$\frac{\boldsymbol{dh_8}}{\boldsymbol{dt}} = \frac{\boldsymbol{W_{TEV}(h_7 - h_8) + Q - P_t}}{\left(\boldsymbol{m_m}\frac{\boldsymbol{C_m}}{\boldsymbol{C_f}} + \boldsymbol{m_f}\right)_{\boldsymbol{ev,sh}}} \tag{5.15}$$

where the subscript *ev,sh* indicates the evaporator shell. The power transferred (P_t) is zero and the rate of energy input (Q) is the heat transfer rate in the evaporator (q_{evap}). Also, from Equation 2.19:

$$\boldsymbol{\tau_{ev,sh}} = \frac{\mathbf{W_{TEV}}}{\left(\frac{\boldsymbol{m_m C_m}}{\boldsymbol{C_f}} + \boldsymbol{m_f}\right)_{\boldsymbol{ev,sh}}} \tag{5.16}$$

Therefore,

$$\frac{\boldsymbol{dh_8}}{\boldsymbol{dt}} = \frac{\boldsymbol{W_{TEV}(h_7 - h_8) + q_{evap}}}{\frac{\boldsymbol{W_{TEV}}}{\boldsymbol{\tau_{ev,sh}}}} \tag{5.17}$$

and

$$\frac{\boldsymbol{dh_8}}{\boldsymbol{dt}} = \boldsymbol{\tau_{ev,sh}(h_7 - h_8)} + \frac{\boldsymbol{\tau_{ev,sh}\, q_{evap}}}{\boldsymbol{W_{TEV}}} \tag{5.18}$$

For a given system and heat load condition, $\tau_{ev,sh}$ is constant, h_7 is a function of P_7, and W_{TEV} is a function of the control inputs n_{TEV} and n_{HGBP}. The rate of heat exchange in the evaporator (q_{evap}) is given by Equations 2.28 and 2.29:

$$\boldsymbol{q_{evap} = \epsilon C_{min}(T_9 - T_7) = W_{fw}(h_9 - h_{10a})} \tag{5.19}$$

For a given chiller, the effectiveness is assumed constant for various flow conditions and the minimum heat capacity (C_{min}) is always that of the fresh water loop. T_9 is a function of T_{10a} and the heat load (HL), given by Equation 5.1. Equilibrium conditions for the system are directly dependent upon heat load conditions and, for a given equilibrium condition, HL is constant. The evaporation temperature (T_7) is a function of the evaporation pressure (P_7).

Then:

$$\boldsymbol{q_{evap} = f(T_{10a}, P_7)} \tag{5.20}$$

Putting these together, Equation 5.18 can be expressed as:

$$\boldsymbol{\frac{dh_8}{dt} = \tau_{ev,sh}(h_7 - h_8) + \frac{\tau_{ev,sh}\epsilon C_{min}\left(T_{10a} + \frac{HL}{(WC_p)_{fw}} - T_7\right)}{W_{TEV}}} \tag{5.21}$$

The right-hand side of this expression is a function of system states and inputs. Thus:

$$\boldsymbol{\frac{dh_8}{dt} = f(h_8, P_7, T_{10a}, n_{TEV}, n_{HGBP})} \tag{5.22}$$

Observe that the functional relationship for dh_8/dt is complicated and would require considerable algebra for linearization. Again, this selection is not unique. If h_1 had been selected as a critical state in Subsection 5.1.1, dh_1/dt would need to be expressed as a function of dh_8/dt. This expression would be non-linear since the flow

rates W_{TEV} and W_{HGBP} are in both the numerator and denominator in Equation 5.13. Thus, the expression for dh_1/dt would prove to be even more complicated than that for dh_8/dt. Furthermore, linearization of dh_1/dt about an equilibrium condition, which is performed in the next subsection, would prove to be much more tedious than the linearization of dh_8/dt. For this reason, h_8 was selected over h_1 to be a critical system state.

Proceeding similarly for the condenser shell exit enthalpy, the state derivative equation for h_4 may be expressed as:

$$\frac{\boldsymbol{dh_4}}{\boldsymbol{dt}} = \boldsymbol{\tau_{co,sh}(h_3 - h_4)} + \frac{\boldsymbol{\tau_{co,sh} W_{sw}(h_{12} - h_{11})}}{\boldsymbol{W_{comp}}} \tag{5.23}$$

$$= \boldsymbol{f(h_4, P_7, n_{TEV}, n_{HGBP}, \omega_{CondPump})}$$

where the subscript *co,sh* denotes the condenser shell. For the fresh water (*fw*) loop, the enthalpy equation is:

$$\frac{\boldsymbol{dh_{10a}}}{\boldsymbol{dt}} = \boldsymbol{\tau_{ev,tu}(h_9 - h_{10a})} + \frac{\boldsymbol{\tau_{ev,tu} \epsilon C_{min}} \left(\boldsymbol{T_{10a}} + \frac{\boldsymbol{HL}}{\boldsymbol{(WC_p)_{fw}}} - \boldsymbol{T_7} \right)}{\boldsymbol{W_{fw}}} \tag{5.24}$$

where the subscript *ev,tu* indicates the tube-side of the evaporator. If we identify (h_o, T_o) as an arbitrary, fixed reference value for water and assume that the specific heat ($C_{p,fw}$) is constant, we observe that:

$$\boldsymbol{h_{10a} - h_o = C_p(T_{10a} - T_o)} \tag{5.25}$$

and

$$\frac{\boldsymbol{d(h_{10a} - h_o)}}{\boldsymbol{dt}} = \frac{\boldsymbol{dh_{10a}}}{\boldsymbol{dt}} = \boldsymbol{C_p} \frac{\boldsymbol{d(T_{10a} - T_o)}}{\boldsymbol{dt}} = \boldsymbol{C_p} \frac{\boldsymbol{dT_{10a}}}{\boldsymbol{dt}} \tag{5.26}$$

Then, using Equations 5.26 and 5.24,

$$\frac{dT_{10a}}{dt} = \tau_{ev,tu}(T_9 - T_{10a}) + \tau_{ev,tu}\epsilon\left(T_{10a} + \frac{HL}{(WC_p)_{fw}} - T_7\right) \tag{5.27}$$

Then, substituting for T_9 from Equation 5.1 and recognizing that the evaporation temperature (T_7) is a function of evaporation pressure (P_7), the state derivative equation for T_{10a}:

$$\begin{aligned}\frac{dT_{10a}}{dt} &= \tau_{ev,tu}\left(\frac{HL}{(WC_p)_{fw}}(1-\epsilon) + \epsilon T_{10a} - \epsilon T_7\right) \\ &= f(T_{10a}, P_7)\end{aligned} \tag{5.28}$$

Finally, consider the rate of change of vapor mass in the evaporator shell with respect to time. By the principle of mass conservation, this is equal to the sum of the flow rate of the vapor entering the shell plus the vapor formed by evaporating the liquid present in the shell minus the flow rate of vapor exiting the shell, i.e.:

$$\frac{dm_v}{dt} = W_{TEV}x_7 + \frac{q_{evap}}{h_{lg}} - W_{TEV} \tag{5.29}$$

Here, $W_{TEV}x_7$ is the fraction of vapor entering the shell and W_{TEV} is the flow rate of vapor exiting the shell. The rate of evaporation of refrigerant liquid is the rate at which heat is transferred to the shell (q_{evap}) relative to the latent heat of evaporation, i.e., the difference between the enthalpy of the saturated liquid and saturated vapor ($h_{lg}=h_l-h_g$). The left-hand side of Equation 5.29 may be expressed as

$$\frac{dm_v}{dt} = \frac{d}{dt}\left(V_v \rho_g\right) = V_v \frac{d\rho_g}{dP_7}\frac{dP_7}{dt} \tag{5.30}$$

where the volume of the vapor in the shell is assumed to be approximately constant. Also, the range for the evaporation pressure across heat load conditions considered here (±25% of the equilibrium condition) is 0.3-0.35 MPa. In this range, the saturated vapor density varies linearly with pressure, i.e., $d\rho_g/dP_7$ is constant. Thus, dP_7/dt can be expressed as:

$$\frac{dP_7}{dt} = \frac{W_{TEV} x_7 + \frac{q_{evap}}{h_{lg}} - W_{TEV}}{V_v \frac{d\rho_g}{dP_7}} \tag{5.31}$$

The time derivative equations of the system states (P_7, h_4, h_8 and T_{10a}) have now been established; these are given by equations 5.31, 5.23, 5.21 and 5.28, respectively. These derivatives are a function of the system states and control inputs. However, these derivatives are highly non-linear. To use linear optimal control theory, it is necessary to linearize the system model about an equilibrium point. This process is discussed in the next subsection.

5.1.3 Linearization about an Equilibrium Point

Now that system states and their time derivatives have been developed, the state of the system at the desired equilibrium point may be determined. From Chapter 2, the default heat load for the York, 200-ton chiller is HL = 709.526 kW. At this point, the values of the control inputs and system states, which collectively describe the complete system state, can be determined at full load. These are provided in Table 5.1 below.

Variable	Value (units)
P_7	328.603 (kPa)
h_4	132.0605 (kJ/kg)
h_8	252.294 (kJ/kg)
T_{10a}	279.82 (K)
n_{TEV}	71.2967 (% open)
n_{HGBP}	56.411 (% open)
$\omega_{CondPump}$	3888.3 (rpm)

Table 5.1: System states and inputs at *HL* = 709.526 kW

To apply optimal control theory, a linearized model of the system about this equilibrium point is required. Define $\boldsymbol{X}$ to be the model state input vector, which is the difference between the system states and their equilibrium values. Similarly, define $\boldsymbol{U}$ to be a model control input vector, which is the difference between the system control inputs and their equilibrium values.

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{P_7 - P_{7eqm}} \\ \boldsymbol{h_4 - h_{4eqm}} \\ \boldsymbol{h_8 - h_{8eqm}} \\ \boldsymbol{T_{10a} - T_{10aeqm}} \end{bmatrix} \tag{5.32}$$

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u_1} \\ \boldsymbol{u_2} \\ \boldsymbol{u_3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{n_{TEV} - n_{TEVeqm}} \\ \boldsymbol{n_{HGBP} - n_{HGBPeqm}} \\ \boldsymbol{\omega_{condpump} - \omega_{condpumpeqm}} \end{bmatrix} \tag{5.33}$$

Note here the difference between system states and control inputs (the variables on the right hand side in these expressions) versus model states and control inputs (the x_i and u_i). System states and inputs are physical variables in the chiller simulation such as evaporation pressure (P_7) and thermostatic expansion valve position ($nTEV$). Model states and inputs are the deviations of system states and inputs about their equilibrium point (x_1, u_1, etc.), and are used in the state-space formulation.

In linearized control theory, the derivative of the model states is expressed as a linear function of the model states and the model inputs, i.e.:

$$\dot{\boldsymbol{X}} = \boldsymbol{AX} + \boldsymbol{BU} \tag{5.34}$$

$$\frac{\boldsymbol{d}}{\boldsymbol{dt}}\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\\ \boldsymbol{x_3}\\ \boldsymbol{x_4}\end{bmatrix} = \boldsymbol{A}\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\\ \boldsymbol{x_3}\\ \boldsymbol{x_4}\end{bmatrix} + \boldsymbol{B}\begin{bmatrix}\boldsymbol{u_1}\\ \boldsymbol{u_2}\\ \boldsymbol{u_3}\end{bmatrix} \tag{5.35}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant coefficient matrices. To obtain linearized model equations, the system state derivative equations, developed in the previous subsection, are reduced to functions of the system states and control inputs, and then linearized about an equilibrium point. This is known as the state-space representation of the system.

For demonstration purposes, what follows is the development of one equation in this state-space formulation, namely the equation for dx_4/dt. The other equations are developed in Appendix B. It is a well-known fact that the evaporation temperature (T_7) and evaporation pressure (P_7) are dependent upon one another; therefore, there is a one-to-one mapping between them. To establish this relationship, begin by using pressure and temperature values obtained from the R-134a data tables in *DTMS:*

$$\frac{\boldsymbol{T_7 - 273.82}}{\boldsymbol{278.18 - 273.82}} = \frac{\boldsymbol{P_7 - 3 * 10^5}}{\boldsymbol{(3.5 - 3) * 10^5}} \tag{5.36}$$

Then:

$$\boldsymbol{T_7 = T_{7j} P_7 + T_{7k}} \tag{5.37}$$

where T_{7j} = 8.72x10^{-5} K/Pa and T_{7k} = 247.66 K. These are then the coefficients of the linearized variation of T_7 with P_7. For the remainder of this chapter, a coefficient of the

linearized variation of a system variable with respect to system states will always contain the subscript *i*, *j,* or *k* to uniquely identify that coefficient.

Substituting for P_7 in terms of x_1 gives:

$$\boldsymbol{T_7 = T_{7j}(P_{7eqm} + x_1) + T_{7k}}$$

$$\boldsymbol{T_7 = T_{7j}x_1 + \ (T_{7j}P_{7eqm} + T_{7k})} \tag{5.38}$$

Inserting this result into Equation 5.28 then produces:

$$\frac{\boldsymbol{1}}{\boldsymbol{\tau_{ev,tu}}}\frac{\boldsymbol{dT_{10a}}}{\boldsymbol{dt}} = \frac{\boldsymbol{1}}{\boldsymbol{\tau_{ev,tu}}}\frac{\boldsymbol{d}}{\boldsymbol{dt}}\left(\boldsymbol{T_{10aeqm} + x_4}\right) = \frac{\boldsymbol{1}}{\boldsymbol{\tau_{ev,tu}}}\frac{\boldsymbol{dx_4}}{\boldsymbol{dt}} \tag{5.39}$$

$$= \frac{\boldsymbol{HL}}{\boldsymbol{C_{fw}}}(\boldsymbol{1-\epsilon}) + \boldsymbol{\epsilon T_{10aeqm} + \epsilon x_4 - \epsilon T_{7j}x_1 + \ (T_{7j}P_{7eqm} + T_{7k})}$$

All variables shown above are constants or states (x_1 and x_4). Thus,

$$\frac{\boldsymbol{dx_4}}{\boldsymbol{dt}} = \boldsymbol{\beta x_1 + \gamma x_4 + \delta} \tag{5.40}$$

where β, γ and δ are constants obtained from Equation 5.39.

In the actual, non-linear system all the equilibrium terms in Equation 5.39 for dx_4/dt would cancel out. However, due to the system linearization, there is some error between the linearized values and their non-linear counterparts. Therefore, these linearized terms do not cancel out. Additionally, there is always a numerical truncation error associated with the accuracy of storing numbers in software. This is minimized by specifying double-precision floating-point numbers in the software. Due to these factors, there is always a small constant residual in the linearized equation for dx_4/dt, in this case δ, which represents the error between the linear and non-linear system. Since this term

does not exist in the non-linear system, and has no physical significance, it is ignored. Therefore, the linearized state derivative equation reduces to:

$$\frac{\boldsymbol{dx_4}}{\boldsymbol{dt}} = \boldsymbol{\beta x_1 + \gamma x_4} \tag{5.41}$$

This is the fourth of four scalar equation in Equation 5.35. Thus, $\beta = \boldsymbol{A}(4,1)$ and $\gamma = \boldsymbol{A}(4,4)$. Obviously, all other elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ in the fourth scalar equation are zero. In this fashion, using the linearization techniques demonstrated above, equations for model state derivatives are obtained, and the coefficients of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are determined, thus fully quantifying the linear system model. Development of the other elements of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are addressed in Appendix A**.**

During linearization, terms frequently arise where the numerator and denominator both require linearization. For example, Equation 5.21 for dh_8/dt has a term, call it α, given by:

$$\boldsymbol{\alpha} = \frac{\boldsymbol{\tau_{ev,sh} \epsilon C_{min}} \left(\boldsymbol{T_{10a}} + \frac{\boldsymbol{HL}}{\boldsymbol{(WC_p)_{fw}}} - \boldsymbol{T_7} \right)}{\boldsymbol{W_{TEV}}} \tag{5.42}$$

From Equation 5.10, W_{TEV} is a function of both n_{TEV} and n_{HGBP}. By plotting W_{TEV} against both n_{TEV} and n_{HGBP} for the range of values that these valve positions might take during a simulation, an equation for W_{TEV} has been developed. This equation is presented as a plane in a 3-dimensional plot (with n_{TEV} and n_{HGBP} as the two independent variables) to obtain a linearized equation for W_{TEV}. This linearization process and its result are described in Appendix A. The end result is an expression of the form:

$$\boldsymbol{W_{TEV} = W_{TEVi} n_{TEV} + W_{TEVj} n_{HGBP} + W_{TEVk}} \quad (5.43)$$

Substituting for n_{TEV} and n_{HGBP} in terms of u_1 and u_2 using Equation 5.33 produces:

$$\boldsymbol{W_{TEV} = W_{TEVi}(n_{TEVeqm} + u_1) + W_{TEVj}(n_{HGBPeqm} + u_2) + W_{TEVk}} \quad (5.44)$$

$$\boldsymbol{= W_{TEVi} u_1 + W_{TEVj} u_2 + (W_{TEV})_{eqm}}$$

Taking the reciprocal results in:

$$\boldsymbol{\frac{1}{W_{TEV}} = \frac{1}{W_{TEVi} u_1 + W_{TEVj} u_2 + (W_{TEV})_{eqm}}} \quad (5.45)$$

$$\boldsymbol{= \frac{1}{(W_{TEV})_{eqm}} * \frac{1}{1 + \left\{\frac{W_{TEVi}}{(W_{TEV})_{eqm}} u_1 + \frac{W_{TEVj}}{(W_{TEV})_{eqm}} u_2\right\}}}$$

Then, using a Taylor series expansion for $(1+z)^{-1}$:

$$\boldsymbol{\frac{1}{1+z} = 1 - z + \frac{z^2}{2!} - \frac{z^3}{3!} + \cdots} \quad (5.46)$$

produces:

$$\boldsymbol{\frac{1}{W_{TEV}} = \frac{1}{(W_{TEV})_{eqm}} \left(1 - \left\{\frac{W_{TEVi}}{(W_{TEV})_{eqm}} u_1 + \frac{W_{TEVj}}{(W_{TEV})_{eqm}} u_2\right\}\right)} \quad (5.47)$$

Substituting this into Equation 5.42 for α:

$$\boldsymbol{\frac{\alpha}{\tau_{ev,sh} \epsilon C_{min}}} \quad (5.48)$$

$$\boldsymbol{= \left(T_{10a} + \frac{HL}{(WC_p)_{fw}} - T_7\right) \frac{1}{(W_{TEV})_{eqm}} \left(1 - \frac{W_{TEVi}}{(W_{TEV})_{eqm}} u_1 - \frac{W_{TEVj}}{(W_{TEV})_{eqm}} u_2\right)}$$

In this equation, T_7 is a function of x_1 from Equation 5.38. Also, T_{10a} is a function of x_4, respectively from Equation 5.32. Thus, substituting for T_{10a} and T_7 in terms of x_4 and x_1 respectively, and expanding the terms in brackets, produces mixed second-order terms of

the form $x_i u_j$. For small deviations in model states and inputs (x and u, respectively), these second-order terms may be ignored since they are much smaller than first-order terms. This is also the case for homogeneous second-order terms such as $x_i x_j$ and x_i^2. The use of this established linearization method produces a linearized system model of the form shown in Equation 5.34. The details are addressed in Appendix A.

5.2 Optimal Control Theory - Linear Quadratic Regulator

The linearized system model developed in the previous section clears the way to apply the principles of optimal control in the chiller model. Formulation of optimal control requires that an objective function (J) be minimized subject to system constraints. J is a scalar indicator of all system and control input deviations about equilibrium wherein appropriate weights are assigned to each input. This weighing process is dependent on the user's preference for weighing equilibrium at one state in relation to others; this process will be explained as this section progresses.

In the case of optimal chiller control, the system must reach a defined equilibrium point at steady-state as determined by the heat load condition. This condition corresponds to all model states and model inputs approaching zero, i.e. $X_{ss} \rightarrow 0$ and $U_{ss} \rightarrow 0$ as $t \rightarrow \infty$. In the vocabulary of control engineering, is called an "infinite horizon" problem. This condition is described by the following formulation of the objective function:

$$\boldsymbol{J = \int_0^{\infty} (X^T Q X + U^T R U) dt} \tag{5.49}$$

where $\boldsymbol{X}$ and $\boldsymbol{U}$ are the model state and model input vectors, respectively. The superscript $\boldsymbol{T}$ indicates a transpose of these vectors. $\boldsymbol{Q}$ and $\boldsymbol{R}$ are constant diagonal weighing matrices appropriate to the particular control application. $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive semi-definite and positive definite diagonal matrices, respectively. For the infinite horizon problem, they are constant matrices. The optimal control problem is then to:

$$\boldsymbol{min\, J} = \int_{\boldsymbol{0}}^{\infty} (\boldsymbol{X^T Q X + U^T R U})\boldsymbol{dt} \qquad (5.50)$$

$$\boldsymbol{subject\ to\ \ \dot{X} = AX + BU}$$

Expanding the integrand in the expression for J above produces:

$$\boldsymbol{J} = \int_{\boldsymbol{0}}^{\infty} (\boldsymbol{\Sigma x_m^2 q_m + \Sigma u_n^2 r_n})\boldsymbol{dt} \qquad (5.51)$$

where q_m and r_n are the indexed diagonal elements of matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, respectively. In this formulation, the integrand is a quadratic function of the linearized model states and inputs. Also, as mentioned earlier, the system reaches steady state as $t \rightarrow \infty$. Therefore, this formulation is called an Infinite Horizon Linear Quadratic Regulator (LQR) problem.

Since J is to be minimized, a higher value for the state-weighing factor (q_m) will ensure a smaller value for the corresponding model state (x_m). Similarly, a higher value for the input-weighing factor (r_n) will ensure a smaller value for the corresponding model input (u_n). Therefore, increasing the weighing factor for a particular model state or input implies more emphasis on that equilibrium value and ensures a lower steady-state error. In the case of the York chiller, the temperature of the fresh water exiting the evaporator (T_{10a}) is more important than the evaporation pressure (P_7). This is because the deviation

in P_7 might be as much as 1-2 kPa and yet the system would be considered to be running smoothly. In contrast, a deviation in T_{10a} beyond 1-2 Kelvins would be a cause for concern. Thus, q_4 (corresponding to x_4 and T_{10a}) in this model is assigned a weighing factor about four orders of magnitude larger than q_1 (corresponding to x_1 and P_7).

Equation 5.50 describes the control formulation for the Infinite Horizon LQR problem. The optimal input ($\boldsymbol{U^o}$) for this problem is expressed as:

$$\boldsymbol{U^o = -KX} \tag{5.52}$$

where the superscript indicates optimal and the feedback gain matrix ($\boldsymbol{K}$) is obtained by solving what is known as the differential Riccati equation. In mathematics, a Riccati equation is an ordinary differential equation that is quadratic in the unknown variable. In LQR theory, this equation is given by:

$$\boldsymbol{A^T S(t) + S(t)A - S(t)BR^{-1}B^T S(t) + Q = \frac{dS(t)}{dt}} \tag{5.53}$$

where ***S(t)*** is called the Riccati matrix. Other quantities in this expression have already been defined. The Riccati equation is a purely mathematical formulation used to solve the minimization problem. Since ours is an infinite horizon problem, the right-hand side is zero at steady state, and this equation reduces to the algebraic Riccati equation:

$$\boldsymbol{A^T S_o + S_o A - S_o BR^{-1}B^T S_o + Q = 0} \tag{5.54}$$

where $\boldsymbol{S_o}$ is the steady-state Riccati matrix. The gain matrix may then be expressed as:

$$\boldsymbol{K = R^{-1}B^T S_o} \tag{5.55}$$

Using Equations 5.50, 5.52 and 5.53, the optimal control output based on the given system model and the model states may be calculated at any given time. Because it

was already available, the `lqr()` function in *MATLAB* was used to input the system model (matrices $\boldsymbol{A}$ and $\boldsymbol{B}$) and objective function matrices (matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$), and to solve the steady-state Riccati equation for the output matrices, $\boldsymbol{S_o}$ and $\boldsymbol{K}$.

The resulting gain matrix is then used in *DTMS* to create a new control class named *ChillerOptimalControl*. This class has access to the various chiller components and states using the internally defined `set()` and `get()` functions. Using the formulation for the four system states and the value of the feedback gain matrix ($\boldsymbol{K}$), the control algorithm calculates $\boldsymbol{U}$, the three control outputs. These outputs for the TEV position, HGBP valve position, and condenser pump speed are then assigned to the three values in $\boldsymbol{U}$ using the `set()` and `get()` functions.

To extend this formulation to other heat load cases, the following steps must be performed manually. System linearization is again performed using the new value of *HL*. Only three variables change with a new value of *HL*. These are the time constants $\tau_{ev,sh}$, $\tau_{ev,tu}$ and $\tau_{co,sh}$. The user may then obtain the new equilibrium value for the system states and inputs by performing a PID simulation in *DTMS*, without changing the control parameters. Then with the new *HL* and new time constant variables and equilibrium values for system states and inputs, one can readily obtain a model linearized about the new equilibrium point. Finally, with the linearized model determined for a new heat load, the scalar objective function (J) is obtained to complete the LQR formulation. The weighing values contained in matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ remain unchanged since the importance of various states relative to each other remains the same. With that the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ have been obtained for the new equilibrium, and the `lqr()` function in *MATLAB*

may be used to calculate the new feedback gain matrix (***K)***. An example of this process is provided later in this chapter.

Using these steps, new feedback gain matrices may be determined for various heat load values. This has already been done and the logic is coded in *DTMS* class *ChillerOptimalControl.* In effect, the feedback matrix has been adjusted based on system conditions. This is known as "gain scheduling".

A procedure now exists to linearize the system model, formulate the objective function, solve for the steady-state feedback gain matrix, and design the optimal controller with the help of *MATLAB*. This procedure need not be followed exactly if the problem formulation is different from the one described in this work, i.e., the chiller control problem. On occasion, the linearized model may work for other circumstances, depending on response sensitivity to various conditions, and gain scheduling may not be required. Some control formulations in *DTMS* may require outputs to have a defined variation over time (called the "tracking problem"). In essence, model-based design of controllers handles each problem according to the characteristics of that problem, i.e., the problem at hand determines the procedures used to design the controller. This is significantly different from previous control methodologies in *DTMS*, such as the PID controller. Therefore, it is important for the control designer to have specific knowledge of optimal control theory, as applied to the particular model under control, before implementing this procedure for controller design.

5.3 STARTUP RESULTS USING LQR CONTROL

This section shows and discusses results obtained by applying the LQR controller formulation to the York 200-ton chiller. As one reads this section, the corresponding behaviors for PID control and no-control presented earlier must be kept in mind. To assist the reader, PID and no-control response are directly co-plotted with optimal control results for comparison purposes. In the graphics that follow, these three control responses are color-coded as **No-Control**, **PID Control** and **Optimal Control**.

Figure 5.2 shows the time response for the fresh water exit temperature in the evaporator (T_{10a}) – the most important monitored variable in these simulations. In all three cases, but not apparent in the figures, there is a spike in the first few seconds of the simulation due to numerical startup response. This affect is caused by adjustment to initial conditions internal to the solver. A detailed explanation of this phenomenon was provided in Chapter 2 and is not repeated here.

In Figure 5.2, it is observed that the no-control simulation falls slowly to the steady-state condition over a lengthy period exceeding 500 seconds, a number which is dependent on the system dynamics. Undershoot is observed in the case of the PID controller in the first 100 seconds. This behavior is absent in the case of optimal controller, even though both are approaching the set point at a similar rate. The PID undershoot may be reduced by increasing derivative control. However this ploy would have the negative effect of producing larger steady-state time response and a larger settling time. Undershoot using the optimal controller is negligible. This occurs in a

physical sense because the controller is aware of, and able to adjust to, all critical system states throughout the simulation.

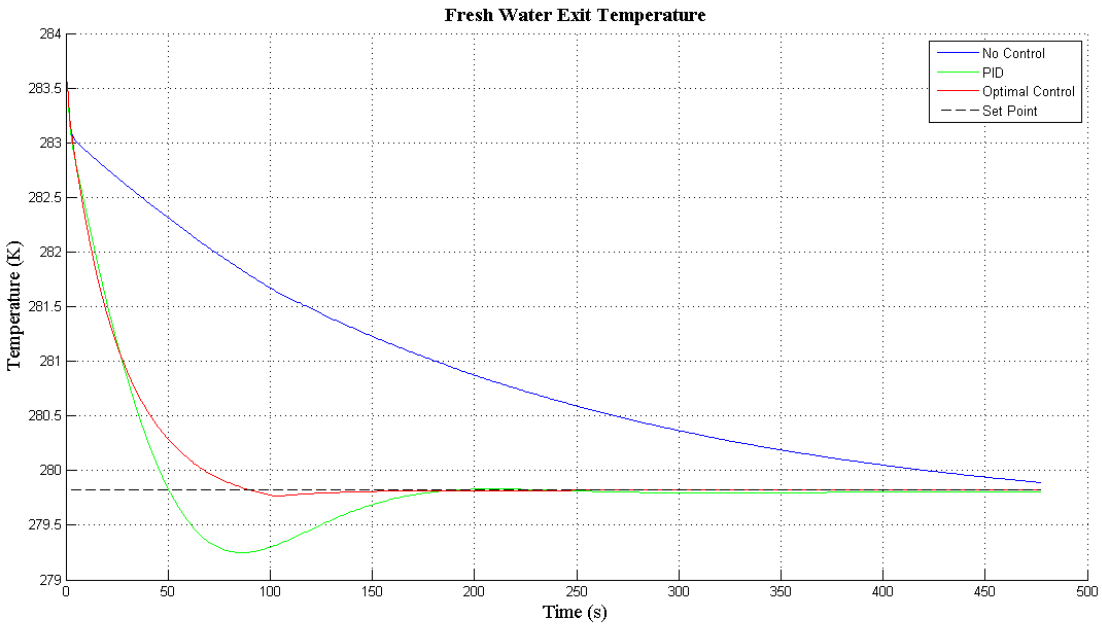

Figure 5.2: Fresh water temperature response for PID, Optimal, and No-Control

Figure 5.3 shows the time response of the pressure in the evaporator for this simulation. As with the fresh water temperature, the evaporation pressure slowly falls to the equilibrium value in the case of no-control. The optimal control, once again, has a negligible undershoot and faster time response compared to the PID control, which again has a significant undershoot and an extended settling time.

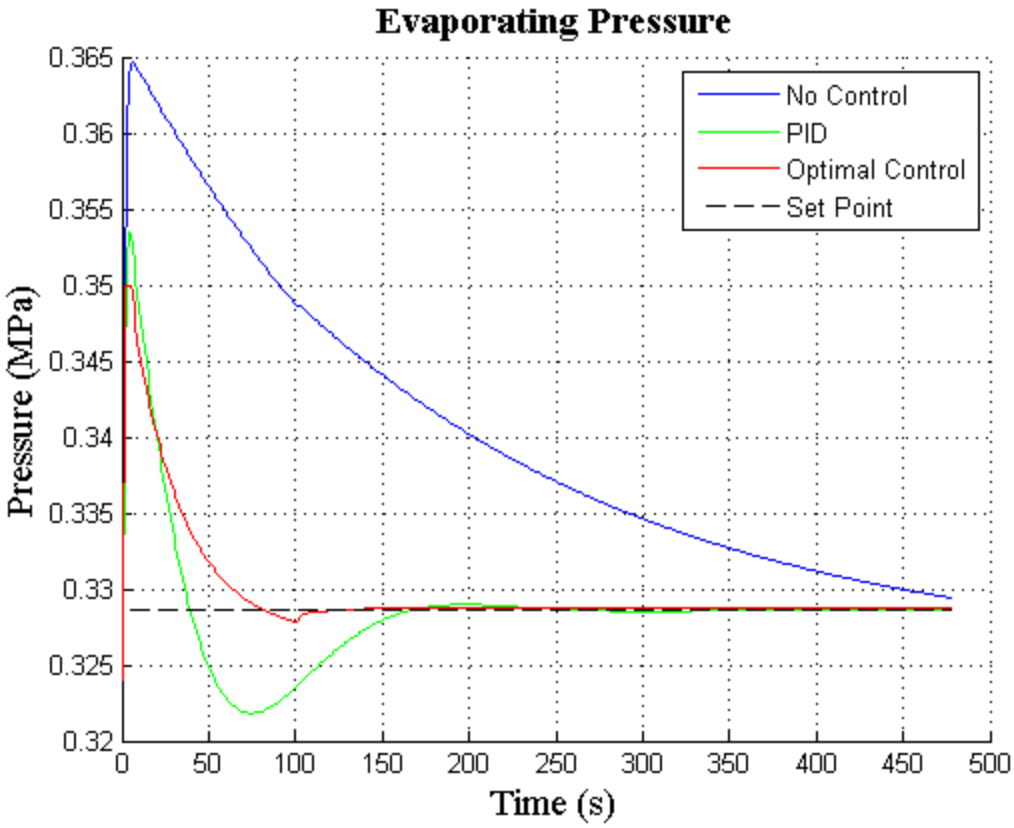

Figure 5.3: Comparison of evaporation pressure for PID, Optimal, and No-Control

Control outputs for the TEV, HGBP valve, and the condenser pump for all three cases are shown in Figures 5.4 and 5.5 below. The control outputs for no-control are maintained constant since none of the control parameters are manipulated in this case. In contrast, there are significant fluctuations in the valve positions for the PID case; the changes in control parameters in this case are large enough to cause extreme and prolonged control outputs. In fact, the HGBP valve control spikes at about 25 seconds and remains fully open for approximately 75 seconds. Adjusting the overall PID control parameters might reduce this fluctuation, but this would also influence the overall settling time for the system. Significant variation in optimal controller output occurs in the first

few seconds when numerical fluctuations dominate. The response then quickly settles to equilibrium conditions in a well-behaved manner. In both cases, TEV control outputs are much better behaved than HGBP valve response.

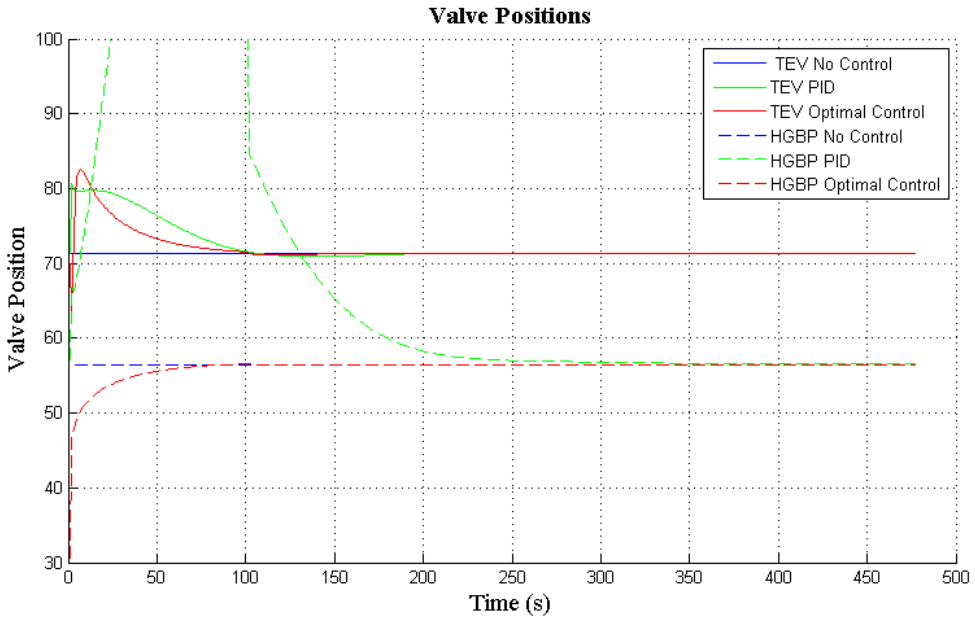

Figure 5.4: Comparison of valve position for PID, Optimal, and No-Control

A similar behavior is seen in Figure 5.5, which displays the time response of the condenser pump speed to controller inputs. Recall that the equilibrium speed is 3888.3 rpm. In the PID case, the pump speed ramps up from about 1000 rpm. This occurs because the PID speed controller only monitors the exit enthalpy of the condenser shell (h_4). As explained in Chapter 2, this enthalpy is initially significantly different from its equilibrium value because the initial refrigerant condition is that of the compressor inlet. This significant negative error causes a control output that lies at the extreme of the pump speed range. As h_4 approaches equilibrium so does the speed of the pump, with a

significant overshoot and slow response time. In contrast to this behavior, the speed output in the optimal control case is based on all four state errors. At the initial condition the error in h_4 is large, but the errors in P_7, h_8, and T_{10a} are small. Also, h_4 is assigned less weight than T_{10a}, which further reduces its influence on the output. Therefore, the model output for the pump response (u_3) is small and the speed hovers around the equilibrium point as indicated in the inset oval.

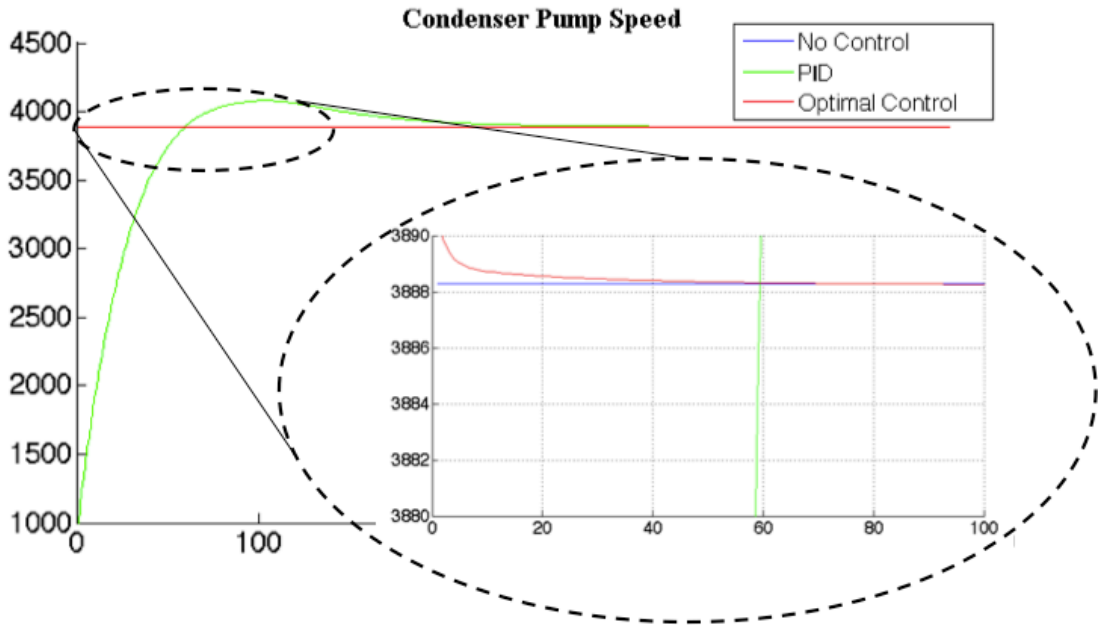

Figure 5.5: Comparison of condenser pump speed for PID, Optimal, and No-Control

Clearly optimal control is a significant improvement over PID control. The most important advantage of optimal control over PID is the global view that the controller has of the linearized system behavior. Because of this, the optimal controller is able to predict and minimize undesirable control outputs to elements of the system that are seemingly distant from the effect of that output.

5.4 System Response to Dynamic Events

Using system startup as a simulation example, it has been shown that the LQR formulation provides the best system response among the cases considered in section 5.3. Also, unlike PID control, this formulation takes into account all system parameters and interactions to provide control outputs that do not cause undesirable effects in parts of the system not directly controlled. Thus, LQR control has proven to be very effective in chiller control.

This section discusses the ability of this control formulation to control the system and provide chilled fresh water at the evaporator tube exit under varying, dynamic, heat load conditions. Figure 5.6 shows the time varying heat load in the fresh water loop used in these simulations. The figure also shows the time varying heat exchange response to this load in the evaporator. During the 500 seconds of simulation the heat load starts at 550 kW, increases to 700 kW (98.6% load), and then decreases to 530 kW (74.7% load). After an initial transient, the actual heat exchanged in the evaporator closely tracks the variation in heat load. The initial transient is a numerical artifact caused by the initial condition of saturated vapor at the evaporator shell inlet, an issue that has been discussed previously. As the shell inlet condition approaches the equilibrium value, it is clearly able to closely follow the time varying heat load. The response time to each step in heat load varies and slight overshoots and undershoots may be observed. These features are capable of being easily quantified.

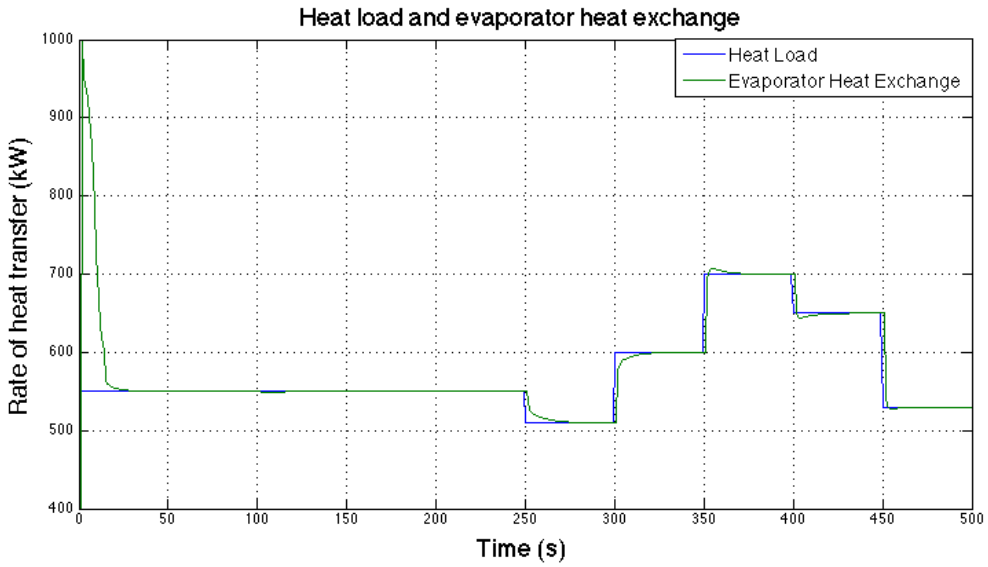

Figure 5.6: Variation of heat load and evaporator response with time

The variation of component exit enthalpies with respect to time is shown in Figure 5.7. Because they are modeled dynamically, the heat exchanger enthalpies vary smoothly in response to the changing load. Also notice the nearly constant value of the compressor suction enthalpy, which is directly controlled by the HGBP valve to give a superheat of 3-5 kJ/kg (or 3-5 K) over the saturated vapor condition. Components such as the TEV, HGBP valve and compressor have small but sudden changes in their enthalpy as the heat load varies. This is because they are not modeled dynamically and, as a result, their exit enthalpy changes almost instantaneously.

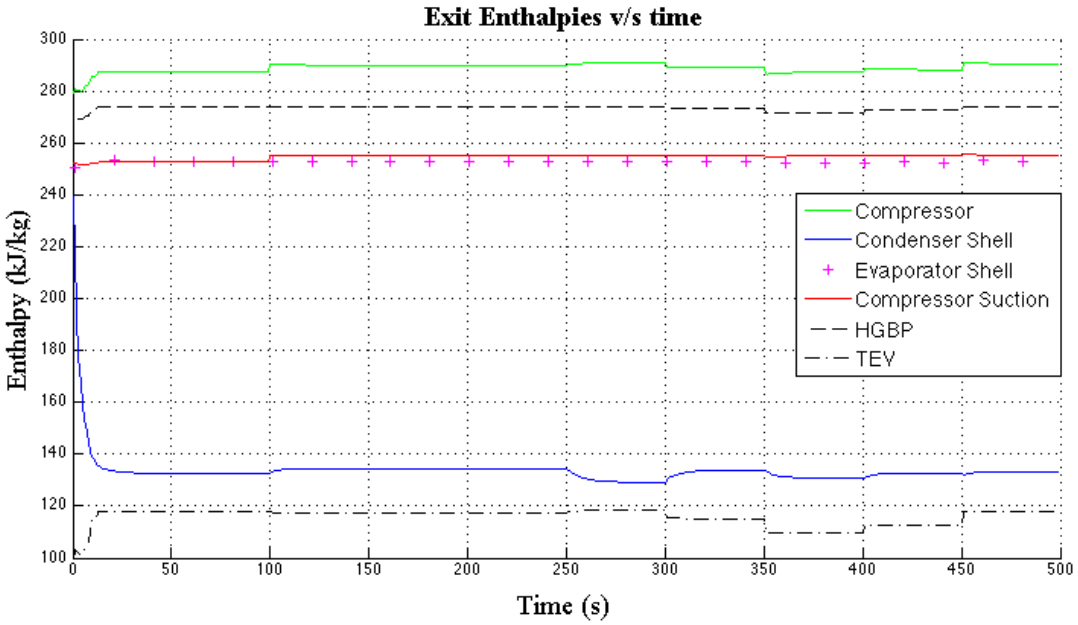

Figure 5.7: Variation of exit enthalpies with time

Figure 5.8 shows the variation of the evaporator fluid temperatures on the shell-side (refrigerant) and tube-side (fresh water). The fresh water exit temperature is a critical system state, and is given the highest preference to reach equilibrium by the LQR controller. Thus, its value remains nearly constant even though the fresh water inlet temperature varies considerably based on the varying heat load condition. The evaporator inlet temperature responds as the fresh water inlet temperature changes, in order to minimize the variation of fresh water exit temperature. The evaporation pressure would also change with the evaporating temperature, and this variation is shown in Figure 5.9.

Clearly, the LQR controller is regulating system response in a favorable way, and the controller can handle dynamic heat loads with highly transient conditions effectively. However, there are limitations to LQR control as well. These are discussed next.

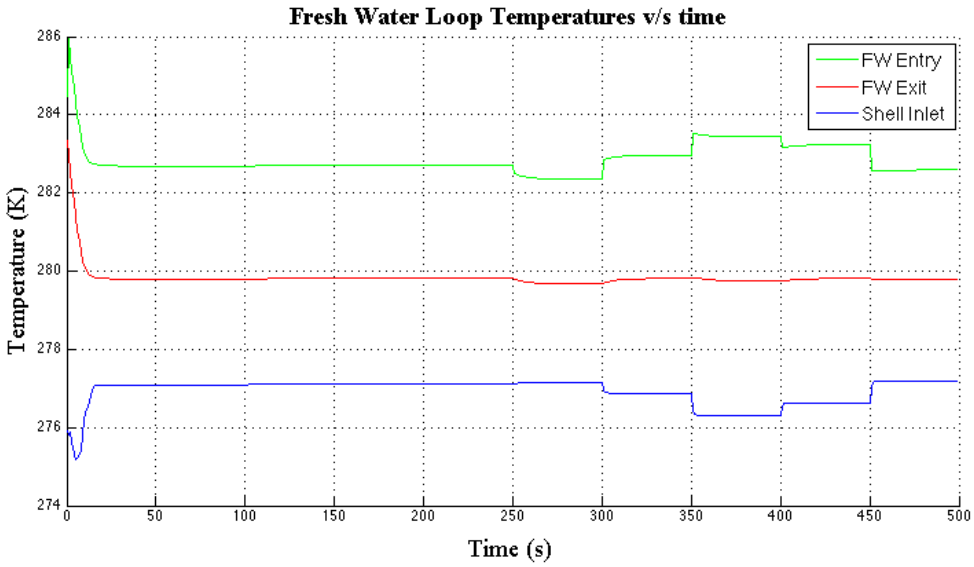

Figure 5.8: Variation of evaporator temperatures with time

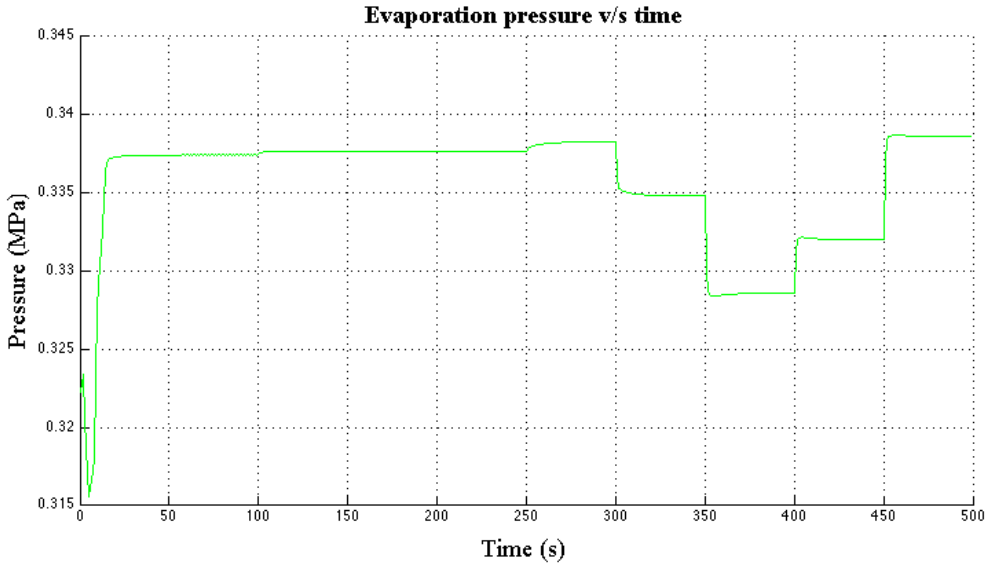

Figure 5.9: Variation of evaporation pressure with time

LQR is a model-based controller design methodology, and thus one must redo the entire procedure if a controller is to be designed for another system, or perhaps even the same system about a significantly different equilibrium point (for example, the chiller system with a different heat load requires re-linearization of the model). Also, if the system is highly non-linear, then linearization may not provide the best results, unless models are built for multiple equilibrium points around the region of non-linearity. This procedure can be very tedious. Thirdly, this system does not account for disturbances of any kind. In reality, disturbances are inherently present in every system, and controlling the system in their presence is important.

These and other issues have been addressed in more advanced control theories, such as Model Predictive Control. The final chapter will put the results and comments about optimal control into perspective, and provide suggestions for further improvements to model simulation and control in *DTMS*.

Chapter 6: Conclusions, Observations, and Future Work

6.1 Conclusions and Observations

For the past decade, the US Navy has committed to fundamental research and technology development on its next generation of surface ships. The vision is that these warships will be dynamically reconfigurable, energy-efficient, and have state-of-the-art pulsed energy weapons and sensors onboard. Therefore, the Navy has focused its attention on integrated electric propulsion and all-electric power distribution, i.e., a huge expansion in dynamic systems powered solely by electricity. This large increase in highly dynamic on-board electrical systems will produce correspondingly large amounts of heat generation which, if not managed properly, will likely produce significant thermal side effects that have the potential to produce catastrophic failures at system and component levels. Thus, shipboard thermal management is considered an enabler for the innovative technologies likely to appear on an all-electric ship (AES).

As a measure to prevent the outcome described above, considerable research is being done on system-level and dynamic thermal management on an AES by the Electric Ship Research and Development Consortium (ESRDC). Because of the large-scale nature of the system-level, dynamic heat loads likely to be encountered on an AES, it would clearly be time and cost prohibitive to conduct experiments to characterize this thermal management environment. Therefore, early in the existence of the ESRDC, it was decided to use simulation techniques with a view to reducing experimental costs and facilitate decision making during the design process.

While commercial software tools were initially used to conduct these system-level simulations, it soon became apparent that these tools were not adequate to address the unique nature of the AES environment. No commercial software available at the time had the ability to handle interdisciplinary thermal-mechanical-electrical simulations of systems with highly transient electric loads and heat generation in a large number of components. Therefore, a decision was made to develop an in-house, highly customizable simulation framework to address thermal management issues across both the mechanical and electrical domains. This software environment is now called the Dynamic Thermal Modeling and Simulation (*DTMS*) framework.

DTMS has grown into a sophisticated software platform capable of modeling and simulating complex physical systems while simultaneously extending its ease of use for both the model developer and simulation user. However, the controller implementation in *DTMS* has always been primitive and inefficient and/or unreliable in handling large-scale system controls. The PID controller that was implemented in *DTMS* at the initiation of the research reported here was localized in nature and, in certain situations, caused undesirable effects in other parts of the system. Therefore, the purpose of this research was to introduce modern control theory into *DTMS* to provide the framework with the ability to control large-scale system simulations.

The research reported in this thesis used control of a marine chiller as a simulation vehicle. After successful construction of and trial simulations with a vapor compression chiller patterned after the York 200-ton marine chiller, several control strategies were implemented. These included the existing and well-established PID controller as well as a

new controller, based on optimal control theory that was introduced into the *DTMS* environment. Results for chiller simulations in the case of no-control, PID control, and optimal control are presented here. The comparative effectiveness of these controls in bringing the chiller to startup equilibrium are investigated. Finally, response of the chiller model and the optimal controller to highly dynamic, varying heat loads was tested.

Initially, models were created in the latest version of *DTMS* to simulate the function of various components that comprise a chiller, i.e., pipes, heat exchangers, thermostatic expansion valves, and a compressor. These components were then connected in a systematic manner to simulate the operation of the York 200-ton chiller, without the application of controls, to obtain an indication of simulation function and accuracy. The equilibrium values of the simulation were compared with the theoretically calculated equilibrium results. These were found to be in agreement, with errors on the order of 1%.

The fundamentals of PID controls were introduced and then implemented in the model of the York chiller. A PID feedback control loop is based on controlling a single variable in the system to cause it to reach a pre-defined set point. The PID controller in *DTMS* is modeled as a special case of the transfer function control scheme. The control output is dependent on three gains defined by the user before the simulation begins. A PID controller is simple to implement but responses are inherently local and multiple controls in a system or subsystem simulation can easily lead to conflicts.

Optimal control theory, as applied to the marine chiller, was then developed and introduced into the DTMS environment. The chiller control problem was modeled as an Infinite Horizon Linear Quadratic Regulator (LQR) problem. This approach requires a

system model that is linearized about a desired equilibrium point. Thus, significant effort is required to obtain system equations and then linearize these about an equilibrium state. However, this formulation is not local and does not create undesirable effects in parts of the system that are not controlled directly by controller inputs. Using the York chiller as an example, specific steps required to formulate the LQR problem are documented in the report. Implementation of the LQR controller was demonstrated for the startup to steady-state function of the chiller at full load. Direct comparison of results are made to the companion PID and no-control cases.

Treatment of the optimal controller ends with simulation of the chiller and its LQR controller under the influence of varying dynamic heat loads in the fresh water loop. The heat load variation examined has highly transient characteristics that affect the temperature of the fresh water entering the chiller, as well as the refrigerant pressure and temperature in the evaporator. The LQR formulation is shown to actively adjust to these varying operating points in a smooth and responsive manner.

These comparisons clearly show that the PID controller simulation exhibits significant undershoot while approaching equilibrium. The LQR controller exhibits negligible undershoot as well as a smooth and rapid approach to equilibrium. Steady-state operation was reached in 125 seconds for the optimal control case, much faster than the 175 seconds with undershoot that was required in the PID case. It was clear that the optimal controller provided the “best” system response.

Also, the system response of the LQR controller under dynamic thermal loading conditions was studied. Heat loads varied from 75% to 100% of the York chiller’s

cooling capacity as step inputs at various times. Despite these variations, the temperature of the fresh water exiting the chiller fluctuated about the set point of 279.82 K with maximum amplitude of 0.15 K. It was concluded that the LQR controller satisfactorily controls the chiller system under these highly transient conditions.

Apart from the full-load case, the optimal controller can also be used for the chiller at partial load conditions. This would imply a new operating equilibrium point for the chiller, and re-linearization of the system model about this point. However, this process is not as tedious as it may appear, and the procedural steps to achieve this are provided. Thus, depending on the magnitude of the dynamic heat load, various control outputs may be developed that cause the system to reach various operating points. This process is known as "gain scheduling", and logic has been incorporated into the optimal control methodology in *DTMS* to achieve this. Gain scheduling, combined with optimal control theory, can then be used to control the system at various operating conditions.

The specific procedure to formulate an LQR controller for a system other than a chiller can be extracted from the examples provided in this report. This process requires more effort and understanding than for a simple PID implementation. However, the technique is state-of-the art and would be expected to provide better response than PID control in most cases. The control developer must make the call on choosing the simplicity of classical control approaches such as PID or on/off control, versus the more complicated, and more elegant, structures of modern control theory.

6.2 RECOMMENDATIONS FOR FUTURE WORK

DTMS is now equipped with an LQR formulation for the York 200-ton chiller model. The formulation can also be extended to other systems by using the steps documented in this report. However, there are more complicated, but often better approaches to system control. Formulations such as the Linear Quadratic Gaussian control can account for uncertain linear systems (or systems linearized about an equilibrium point) disturbed by white noise. Model Predictive control is an advanced method of process control, that relies on a dynamic model of the process, the history of previous system inputs, and an optimization function to minimize the error within a limited period of time, i.e., the "finite horizon" problem. These and other control techniques may be considered for improved system response and enhanced accuracy when modeling uncertainties in a system, subject to the desired trade off between controller complexity and improved system response.

There are better, and more complicated, approaches to modeling systems that can be incorporated into *DTMS*. Future work might include more detailed models of the components that comprise a marine chiller. Finally, to make *DTMS* a viable option to model even larger and more complex systems, which is definitely a consideration when ship system-level simulations are desired, it would be beneficial to have a graphical user-interface (GUI) in *DTMS* that the end-user can interact with to create large-scale models. This GUI might implement automatic code-generation to assist the user in creating simulation code.

Appendix A: System Identification

This appendix contains information on obtaining system equations from simulation data. This is needed primarily because the fluid solvers solve for the pressures and flow numerically, since the derivation of analytical solutions for these is complicated. What follows is the derivation of the flowrates W_{TEV} and W_{HGBP} in terms of system inputs n_{TEV} and n_{HGBP}.

The first step in system identification is to obtain variable and input data at discrete points throughout the operating range of the system. This includes steady state as well as transient values. For example, for the full heat load case of 709.526 kW, the position of the TEV varies in the range 70-80% open and that of the HGBP valve varies in the range 50-100% open. Thus, *DTMS* simulations were conducted to obtain the flowrates for certain fixed valve postions in this range, namely (n_{TEV}, n_{HGBP}) = {(70,50), (70,70), (70,90), (80,70), (80,80), (80,90), (80, 100), (71.3, 56.42)} where the last data point is that of the equilibrium state for this heat load. The input values (n_{TEV} and n_{HGBP}) as well as the flowrates (W_{TEV} and W_{HGBP}) obtained through these simulations were stored in *MATLAB* as arrays *PosTEV*, *PosHGBP*, *WTEV*, and *WHGBP* respectively. To get a linear equation of the flowrate in terms of valve inputs, it was required to fit a plane onto the data points, using regression analysis. This equation would be of the form given in Equation 5.43

$$\boldsymbol{W_{TEV} = W_{TEVi} n_{TEV} + W_{TEVj} n_{HGBP} + W_{TEVk}} \quad \text{(A.1)}$$

where the subscripts i, j and k indicate coefficients of linearized variation of W_{TEV} with respect to n_{TEV} and n_{HGBP}. In the case of W_{TEV}, the following *MATLAB* code was used to identify these three coefficients and obtain the maximum error between the actual data point and plane equation obtained.

```
X = [ones(size(PosTEV)) PosTEV PosHGBP];
% data rows in the form [1 nTEV nHGBP]

aTEV = X\WTEV; %Obtain coefficients via regression

Y = X*aTEV; %Obtain WTEV from plane equation
PercentErrorTEV = (Y-WTEV)./WTEV*100; %Calculate
percent error
MaxPercentageErrTEV = max(PercentErrorTEV); %Calculate
maximum percent error
```

Here, W_{TEVk} = `aTEV(1)`, W_{TEVi} = `aTEV(2)` and W_{TEVj} = `aTEV(3)`. Similar code was used to obtain the equation of the W_{HGBP} plane fit on the data points.

$$\boldsymbol{W_{HGBP} = W_{HGBPi} n_{TEV} + W_{HGBPj} n_{HGBP} + W_{HGBPk}} \quad \text{(A.2)}$$

The maximum percentage error within the operating range was 8.25% for W_{TEV} and 9.09% for W_{HGBP}. These errors are considered to be within reasonable limits.

Similarly, the sea water flow rate is expressed as a function of the pump speed and is of the form:

$$\boldsymbol{W_{sw} = W_{swj} \omega_{CondPump} + W_{swk}} \quad \text{(A.3)}$$

Since some system variables are obtained numerically, system variables that depend on these quantities also cannot be obtained analytically. For example, Equation 5.8 states that

$$\boldsymbol{h_7 = h_l(P_4) = h_l\big(f(P_7)\big)} \tag{A.4}$$

Here, P_4 is a function of P_7, but their quantitative relationship is not analytically derivable. Thus, it was necessary to use system identification to obtain h_7 in terms of P_7 as well. Data points were obtained across various operating points in the chiller to use for regression analysis. *MATLAB* code similar to the one above is used to calculate the coefficients of linearized variation of h_7 with respect to P_7. It should be noted that the maximum percentage error for all of the system variables was less than that of W_{HGBP} (i.e. 9.09 %).

Appendix B: System Linearization

Section 5.1 contained information on methods of system linearization about an equilibrium point to obtain the model state derivative equations. One such derivative equation, for dx_4/dt, was obtained in Section 5.1.3. Described below are the derivations of the state derivative equation for the other model states and the determination of matrices ***A*** and ***B*** in the linearized system model. For clarity, the chiller schematic from Figure 5.1 is repeated below.

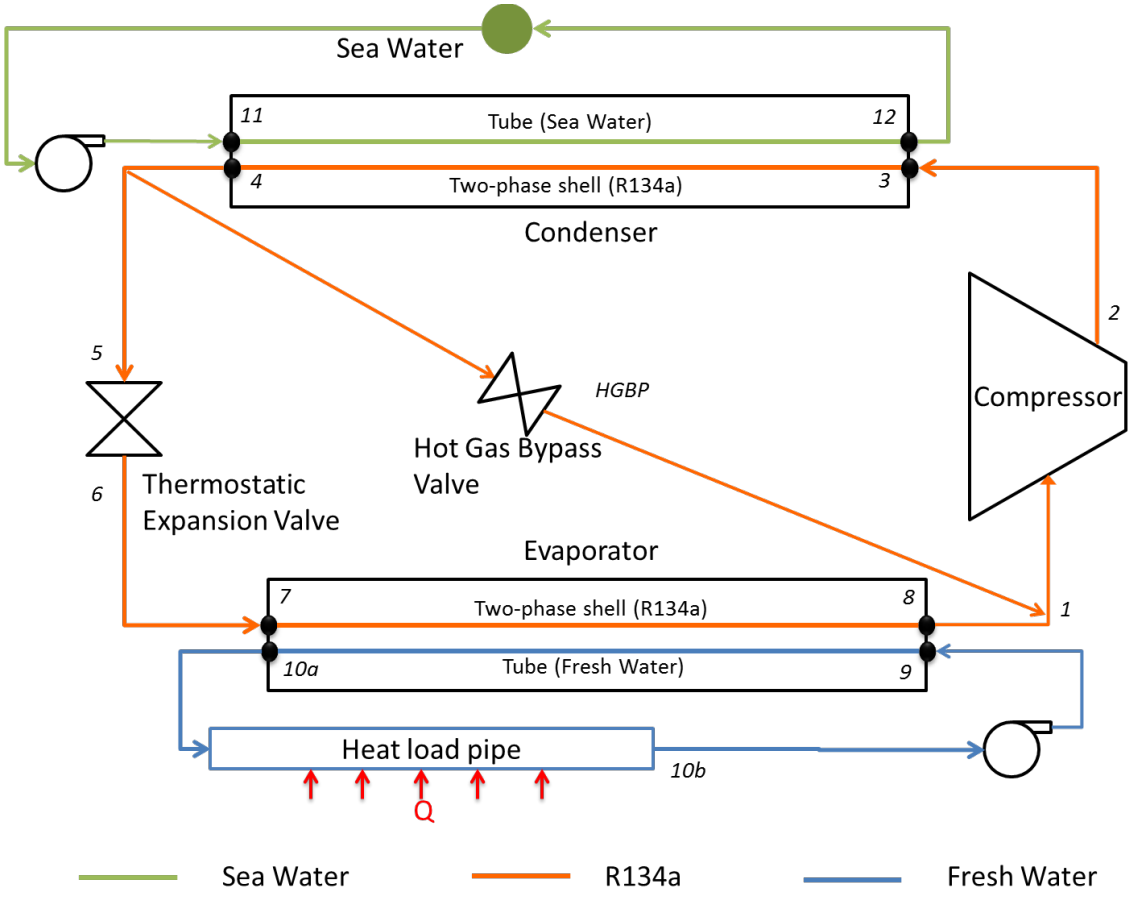

Figure B.1: Schematic of chiller

From chapter 5, the system state derivative equation for T_{10a} and the model state derivative for equation for x_4 are given below

$$\frac{\mathbf{1}}{\boldsymbol{\tau_{ev,tu}}}\frac{\boldsymbol{dT_{10a}}}{\boldsymbol{dt}} = \frac{\boldsymbol{HL}}{\boldsymbol{C_{fw}}}(\mathbf{1}-\boldsymbol{\epsilon}) + \boldsymbol{\epsilon T_{10aeqm}} + \boldsymbol{\epsilon x_4} - \boldsymbol{\epsilon T_{7j} x_1} + (\boldsymbol{T_{7j} P_{7eqm}} + \boldsymbol{T_{7k}}) \tag{B.1}$$

$$\frac{\boldsymbol{dx_4}}{\boldsymbol{dt}} = \boldsymbol{\beta x_1} + \boldsymbol{\gamma x_4} \tag{B.2}$$

where β and γ are constants and the subscript *ev,tu* indicates the evaporator tube. These constants were explained in Section 5.1.3 to be elements of matrix $\boldsymbol{A}$. Also, the equations relating the model states and inputs to the system states and inputs will be used frequently, and are repeated here:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \boldsymbol{x_4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{P_7 - P_{7eqm}} \\ \boldsymbol{h_4 - h_{4eqm}} \\ \boldsymbol{h_8 - h_{8eqm}} \\ \boldsymbol{T_{10a} - T_{10aeqm}} \end{bmatrix} \tag{B.3}$$

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u_1} \\ \boldsymbol{u_2} \\ \boldsymbol{u_3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{n_{TEV} - n_{TEVeqm}} \\ \boldsymbol{n_{HGBP} - n_{HGBPeqm}} \\ \boldsymbol{\omega_{condpump} - \omega_{condpumpeqm}} \end{bmatrix} \tag{B.4}$$

To obtain the model state derivative equation for x_2, consider the corresponding system state derivation for h_4, given by Equation 5.23.

$$\frac{\boldsymbol{dh_4}}{\boldsymbol{dt}} = \boldsymbol{\tau_{co,sh}}(\boldsymbol{h_3 - h_4}) + \frac{\boldsymbol{\tau_{co,sh} W_{sw}}(\boldsymbol{h_{12} - h_{11}})}{\boldsymbol{W_{comp}}} \tag{B.5}$$

where the symbols have their usual meanings, the subscript *co,sh* indicates condenser shell, *comp* indicates compressor and *sw* indicates seawater. W_{sw} is expressed as a linear function of $\omega_{CondPump}$ from Equation A.3. The seawater is assumed to enter from a reservoir and exit into the reservoir, making h_{12} and h_{11} fixed. W_{comp} is the flowrate

through the compressor. This is simply the sum of the flow rates through the TEV and the HGBP valve, given by Equations A.1 and A.2.

$$\boldsymbol{W_{comp} = W_{TEV} + W_{HBGP}} \tag{B.6}$$
$$\boldsymbol{W_{comp} = W_{TEVi}u_1 + W_{TEVj}u_2 + (W_{TEV})_{eqm} + W_{HGBPi}u_1 + W_{HGBPj}u_2 + (W_{HGBP})_{eqm}}$$

Therefore,

$$\boldsymbol{W_{comp} = W_{compi}u_1 + W_{compj}u_2 + \left(W_{comp}\right)_{eqm}} \tag{B.7}$$

Now, to linearize the term ($1/W_{comp}$), which appears in Equation B.5, express it as:

$$\frac{\boldsymbol{1}}{\boldsymbol{W_{comp}}} = \frac{\boldsymbol{1}}{\boldsymbol{W_{compi}u_1 + W_{compj}u_2 + \left(W_{comp}\right)_{eqm}}} \tag{B.8}$$
$$= \frac{\boldsymbol{1}}{\boldsymbol{\left(W_{comp}\right)_{eqm}}} \frac{\boldsymbol{1}}{\left\{\boldsymbol{1 + \frac{W_{compi}}{(W_{comp})_{eqm}}u_1 + \frac{W_{compj}}{(W_{comp})_{eqm}}u_2}\right\}}$$

Use the Taylor series expansion for $(1+z)^{-1}$:

$$\frac{\boldsymbol{1}}{\boldsymbol{1+z}} = \boldsymbol{1 - z + \frac{z^2}{2!} - \frac{z^3}{3!} + \cdots} \tag{B.9}$$

to get the linearized form:

$$\frac{\boldsymbol{1}}{\boldsymbol{W_{comp}}} = \frac{\boldsymbol{1}}{\boldsymbol{\left(W_{comp}\right)_{eqm}}} \left\{\boldsymbol{1 - \frac{W_{compi}}{\left(W_{comp}\right)_{eqm}}u_1 - \frac{W_{compj}}{\left(W_{comp}\right)_{eqm}}u_2}\right\} \tag{B.10}$$

The exit enthalpy of the compressor (h_3) is dependent on the inlet pressure (P_1), which is function of the evaporation pressure (P_7). Thus, similar to h_7, its relationship is obtained by system identification, as explained in Appendix A.

$$\boldsymbol{h_3 = h_{3j}P_7 + h_{3k}} \tag{B.11}$$

Substituting for h_3, $1/W_{comp}$ and W_{sw} in Equation B.5 produces:

$$\frac{\boldsymbol{dh_4}}{\boldsymbol{dt}} = \boldsymbol{\tau_{co,sh}}\left(\boldsymbol{h_{3j}P_7 + h_{3k} - h_4}\right) + \frac{\boldsymbol{\tau_{co,sh}}\left(\boldsymbol{W_{swj}\omega_{CondPump} + W_{swk}}\right)\left(\boldsymbol{h_{12} - h_{11}}\right)}{\left(\boldsymbol{W_{comp}}\right)_{\boldsymbol{eqm}}} * \left\{\boldsymbol{1 - \frac{W_{compi}}{\left(W_{comp}\right)_{eqm}}u_1 - \frac{W_{compj}}{\left(W_{comp}\right)_{eqm}}u_2}\right\} \tag{B.12}$$

This equation consists of system states (P_7, h_4), system inputs ($\omega_{CondPump}$), model inputs (u_1, u_2) and constants. Substitution of the system states and inputs in term so of model states and inputs is performed using Equations B.3 and B.4. This transforms the equation in a function of model states and input (x_1, x_2, u_1, u_2, u_3), of the form:

$$\frac{\boldsymbol{dx_2}}{\boldsymbol{dt}} = \boldsymbol{\beta_1(\beta_2 x_1 - x_2 + \beta_3)} + \boldsymbol{\beta_4(\beta_5 u_3 + \beta_6)\{1 + \beta_7 u_1 + \beta_8 u_2\}} \tag{B.13}$$

where β_is are all constants. On expanding the brackets, terms of the form u_iu_j are ignored. Also, the constant residual ($\beta_1\beta_3 + \beta_4\beta_6$) is an output of linearization error and is ignored. Again, the explanation for ignoring these terms is given in Chapter 5 and will not be repeated. Therefore, the equation above reduces to the form:

$$\frac{\boldsymbol{dx_2}}{\boldsymbol{dt}} = \boldsymbol{\beta_9 x_1 + \beta_{10} x_2 + \beta_{11} u_1 + \beta_{12} u_2 + \beta_{13} u_3} \tag{B.14}$$

This is the second of the four scalar equations in Equation 5.35. Thus, $\beta_9 = \boldsymbol{A}(2,1)$, $\beta_{10} = \boldsymbol{A}(2,2)$, $\beta_{11} = \boldsymbol{B}(2,1)$, $\beta_{12} = \boldsymbol{B}(2,2)$, and $\beta_{13} = \boldsymbol{B}(2,3)$. Other elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ in the second row are zero.

Finally, it is necessary to obtain the model state derivative equation for x_3, which is derived from Equation 5.21.

$$\frac{dh_8}{dt} = \tau_{ev,sh}(h_7 - h_8) + \frac{\tau_{ev,sh}\epsilon C_{min}\left(T_{10a} + \frac{HL}{(WC_p)_{fw}} - T_7\right)}{W_{TEV}} \tag{B.15}$$

where the symbols and subscripts have their usual meanings. The second term on the right-hand side of the equation above was represented as α from Equation 5.42

$$\alpha = \frac{\tau_{ev,sh}\epsilon C_{min}\left(T_{10a} + \frac{HL}{(WC_p)_{fw}} - T_7\right)}{W_{TEV}} \tag{B.16}$$

The linearized version of α was presented in Equation 5.48

$$\frac{\alpha}{\tau_{ev,sh}\epsilon C_{min}} = \left(T_{10a} + \frac{HL}{(WC_p)_{fw}} - T_7\right)\frac{1}{(W_{TEV})_{eqm}}\left(1 - \frac{W_{TEVi}}{(W_{TEV})_{eqm}}u_1 - \frac{W_{TEVj}}{(W_{TEV})_{eqm}}u_2\right) \tag{B.17}$$

It was also explained that T_7 is a linear function of x_1 from Equation 5.38 and that T_{10a} is a function of x_4 from Equation B.3. Therefore:

$$\frac{\alpha}{\tau_{ev,sh}\epsilon C_{min}} = \left(T_{10aeqm} + x_4 + \frac{HL}{(WC_p)_{fw}} - T_{7j}x_1 + \left(T_{7j}P_{7eqm} + T_{7k}\right)\right) * \frac{1}{(W_{TEV})_{eqm}}\left(1 - \frac{W_{TEVi}}{(W_{TEV})_{eqm}}u_1 - \frac{W_{TEVj}}{(W_{TEV})_{eqm}}u_2\right) \tag{B.18}$$

Thus, α is now expressed in terms of the model states and inputs, of the form:

$$\alpha = \gamma_1(\gamma_2 x_1 + x_4 + \gamma_3)(1 - \gamma_4 u_1 - \gamma_5 u_2) \tag{B.19}$$

where γ_is are all constants. Hence, Equation B.30 reduces to

$$\frac{dh_8}{dt} = \tau_{ev,sh}(h_7 - h_8) + \gamma_1(\gamma_2 x_1 + x_4 + \gamma_3)(1 - \gamma_4 u_1 - \gamma_5 u_2) \tag{B.20}$$

The system state h_8 is function of x_3 from Equation B.3, and h_7 is a function of P_7, which is a function of x_1, again from Equation B.3. Substituting for these gives:

$$\frac{\boldsymbol{d(x_3 + h_{8eqm})}}{\boldsymbol{dt}} = \boldsymbol{\tau_{ev,sh}(h_{7j}P_7 + h_{7k} - x_3 - h_{8eqm})}$$
$$\boldsymbol{+\gamma_1(\gamma_2 x_1 + x_4 + \gamma_3)(1 - \gamma_4 u_1 - \gamma_5 u_2)}$$
$$\frac{\boldsymbol{dx_3}}{\boldsymbol{dt}} = \boldsymbol{\tau_{ev,sh}(h_{7j}(x_1 + P_{7eqm}) + h_{7k} - x_3 - h_{8eqm})} \quad \text{(B.21)}$$
$$\boldsymbol{+\gamma_1(\gamma_2 x_1 + x_4 + \gamma_3)(1 - \gamma_4 u_1 - \gamma_5 u_2)}$$

which simplifies to

$$\frac{\boldsymbol{dx_3}}{\boldsymbol{dt}} = \boldsymbol{\gamma_6 x_1 + \gamma_7 x_4 + \gamma_8} \quad \text{(B.22)}$$
$$\boldsymbol{+\gamma_1(\gamma_2 x_1 + x_4 + \gamma_3)(1 - \gamma_4 u_1 - \gamma_5 u_2)}$$

Thus, by ignoring terms of the form $u_i x_j$ and the constant, this equation reduces to:

$$\frac{\boldsymbol{dx_3}}{\boldsymbol{dt}} = \boldsymbol{\gamma_9 x_1 + \gamma_{10} x_4 + \gamma_{11} u_1 + \gamma_{12} u_2} \quad \text{(B.23)}$$

This is the third of four scalar equations in Equation 5.35. Thus, $\gamma_9 = \boldsymbol{A}(3,1)$, $\gamma_{10} = \boldsymbol{A}(3,2)$, $\gamma_{11} = \boldsymbol{B}(3,1)$ and $\gamma_{12} = \boldsymbol{B}(3,2)$. Other elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ in the third row are set to zero.

Another process of linearization involves the use of partial derivatives about the equilibrium point to obtain a model state derivative equation. This method was not used in the research conducted for this thesis. However, for the purpose of demonstration, it is used here for the linearization of dx_1/dt, starting from the corresponding state derivative equation of dP_7/dt, given by Equation 5.31.

$$\frac{\boldsymbol{dP_7}}{\boldsymbol{dt}} = \frac{\boldsymbol{W_{TEV}(x_7 - 1) + \frac{q_{evap}}{h_{lg}}}}{\boldsymbol{V_v \frac{d\rho_g}{dP_7}}} \quad \text{(B.24)}$$

where the volume of vapor in the shell is assumed to be approximately constant, and the value of $d\rho_g \backslash dP_7$ was shown to be constant for the range of evaporation pressure (P_7) considered.

The flowrate W_{TEV} is a function of inputs n_{TEV} and n_{HGBP}, from Equation 5.43.

$$\boldsymbol{W_{TEV} = W_{TEVi} n_{TEV} + W_{TEVj} n_{HGBP} + W_{TEVk}} \tag{B.25}$$

where W_{TEVi}, W_{TEVj} and W_{TEVk} are constants.

To obtain the evaporator shell inlet refrigerant quality (x_7) in terms of model states and inputs, recall that x_7 can be related to the enthalpy h_7, the corresponding saturated liquid enthalpy (h_{l7}) and the latent heat of evaporation (h_{lg7}) as:

$$\boldsymbol{x_7 = \frac{h_7 - h_{l7}}{h_{lg7}}} \tag{B.26}$$

Within the given operating pressure range, the latent heat of evaporation (h_{lg7}) stays approximately constant, varying in the range 196.16 ± 1.5 kJ/kg, a maximum deviation of less than 1%. Thus, h_{lg7} is assumed to have a constant value of 196.16 kJ/kg. Similar to Equation 5.36, variation of h_{l7} with evaporation pressure P_7 can be shown to be linear, and that of h_7 with respect to P_7 is obtained by system identification, explained in Appendix A. Thus:

$$\boldsymbol{h_7 = h_{7j} P_7 + h_{7k}} \tag{B.27}$$

$$\boldsymbol{h_{l7} = h_{l7j} P_7 + h_{l7k}} \tag{B.28}$$

By using these expressions, the refrigerant inlet quality can now be expressed as:

$$\boldsymbol{x_7 = \frac{(h_{7j} P_7 + h_{7k}) - (h_{l7j} P_7 + h_{l7k})}{196.16}} \tag{B.29}$$

implying that x_7 is a linear function of P_7:

$$\boldsymbol{x_7 = x_{7j} P_7 + x_{7k}} \tag{B.30}$$

Equation 5.19 gives the expression for heat exchanged by the evaporator (q_{evap}).

$$\boldsymbol{q_{evap} = \epsilon C_{min}(T_9 - T_7)} \tag{B.31}$$

Since the refrigerant in the shell-side is evaporating, its heat capacity is infinite. Thus:

$$\boldsymbol{C_{min} = C_{ev,tu} = \left(WC_p\right)_{fw}} \tag{B.32}$$

From Equation 5.1, T_9 is a function of T_{10a} and heat load HL, which is repeated here:

$$\boldsymbol{T_9 = T_{10a} + \frac{HL}{\left(WC_p\right)_{fw}}} \tag{B.33}$$

From Equation 5.37, T_7 is a function of the evaporation pressure, P_7.

$$\boldsymbol{T_7 = T_{7j}P_7 + T_{7k}} \tag{B.34}$$

Substituting for T_7, T_9 and C_{min} in Equation B.31 gives:

$$\boldsymbol{q_{evap} = \epsilon\left(WC_p\right)_{fw}\left(T_{10a} + \frac{HL}{\left(WC_p\right)_{fw}} - T_{7j}P_7 - T_{7k}\right)} \tag{B.35}$$

Therefore, substituting for q_{evap}, x_7 and W_{TEV} in Equation B.24 produces:

$$\boldsymbol{V_v\frac{d\rho_g}{dP_7}\frac{dP_7}{dt} = \{W_{TEVi}n_{TEV} + W_{TEVj}n_{HGBP} + W_{TEVk}\}\{x_{7j}P_7 + x_{7k} - 1\}} \tag{B.36}$$
$$\boldsymbol{+\frac{\epsilon\left(WC_p\right)_{fw}}{h_{lg}}\left\{T_{10a} + \frac{HL}{\left(WC_p\right)_{fw}} - T_{7j}P_7 - T_{7k}\right\}}$$

Thus, dP_7/dt is a function of system states (T_{10a}, P_7) and inputs (n_{TEV}, n_{HGBP}).

$$\boldsymbol{\frac{dP_7}{dt} = f(P_7, T_{10a}, n_{TEV}, n_{HGBP})} \tag{B.37}$$

To obtain a linearized form of the above equation, expand it using the Taylor series for multiple variables:

$$\boldsymbol{f(P_7, T_{10a}, n_{TEV}, n_{HGBP}) = f(P_7, T_{10a}, n_{TEV}, n_{HGBP})_{eqm}} \quad \text{(B.38)}$$

$$+\left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial P_7}}\right)_{\boldsymbol{eqm}}\boldsymbol{(P_7 - P_{7eqm})} + \left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial T_{10a}}}\right)_{\boldsymbol{eqm}}\boldsymbol{(T_{10a} - T_{10aeqm})}$$

$$+\left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial n_{TEV}}}\right)_{\boldsymbol{eqm}}\boldsymbol{(n_{TEV} - n_{TEVeqm})} + \left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial n_{HGBP}}}\right)_{\boldsymbol{eqm}}\boldsymbol{(n_{HGBP} - n_{HGBPeqm})}$$

$$+ \ \ldots \ (\boldsymbol{higher\ order\ terms})$$

Here, all terms of the type (V-V_{eqm}) are deviations of the variable V about the equilbrium point. From Equations B.3 and B.4, it is established that these are model states and inputs. Also, the value of dP_7/dt at equilibrium ($f(P_7,T_{10a},n_{TEV},n_{HGBP})_{eqm}$) will be zero, as the system reaches steady state at the equilibrium point. Therefore:

$$\boldsymbol{f(P_7, T_{10a}, n_{TEV}, n_{HGBP}) = 0} \quad \text{(B.39)}$$

$$+\left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial P_7}}\right)_{\boldsymbol{eqm}}\boldsymbol{x_1} + \left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial T_{10a}}}\right)_{\boldsymbol{eqm}}\boldsymbol{x_4}$$

$$+\left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial n_{TEV}}}\right)_{\boldsymbol{eqm}}\boldsymbol{u_1} + \left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial n_{HGBP}}}\right)_{\boldsymbol{eqm}}\boldsymbol{u_2}$$

where the higher order terms have been ignored. These terms will be of the form u_iu_j, u_ix_j, x_ix_j and other terms of third of higher order. For small deviations in model states and inputs (x and u, respectively), these higher-order terms are much smaller than first-order terms, may be thus be ignored.

It is now necessary to obtain the values of the partial derivatives above at the equilibrium point. From Equation B.36,

$$\left(\frac{\partial f}{\partial P_7}\right)_{eqm} = \frac{1}{V_v \frac{d\rho_g}{dP_7}}\left[\{W_{TEVi} n_{TEV} + W_{TEVj} n_{HGBP} + W_{TEVk}\} x_{7j} + \frac{\epsilon (WC_p)_{fw}}{h_{lg}}\{-T_{7j}\}\right]_{eqm} \quad \text{(B.40)}$$

$$\left(\frac{\partial f}{\partial T_{10a}}\right)_{eqm} = \frac{1}{V_v \frac{d\rho_g}{dP_7}}\left[\frac{\epsilon (WC_p)_{fw}}{h_{lg}}\{1\}\right]_{eqm} \quad \text{(B.41)}$$

$$\left(\frac{\partial f}{\partial n_{TEV}}\right)_{eqm} = \frac{1}{V_v \frac{d\rho_g}{dP_7}}\left[W_{TEVi}\{x_{7j} P_7 + x_{7k} - 1\}\right]_{eqm} \quad \text{(B.42)}$$

$$\left(\frac{\partial f}{\partial n_{HGBP}}\right)_{eqm} = \frac{1}{V_v \frac{d\rho_g}{dP_7}}\left[W_{TEVj}\{x_{7j} P_7 + x_{7k} - 1\}\right]_{eqm} \quad \text{(B.43)}$$

The numerical values of these derivatives are obtained by applying the equilibrium values of the system states and inputs. Hence, the linearized version of dP_7/dt is of the form

$$\frac{dP_7}{dt} = f(P_7, T_{10a}, n_{TEV}, n_{HGBP}) = \alpha_1 x_1 + \alpha_2 x_4 + \alpha_3 u_1 + \alpha_4 u_2 \quad \text{(B.44)}$$

P_7 only differs from x_1 by a fixed equilibrium value, from Equation B.3. Thus,

$$\frac{dP_7}{dt} = \frac{d(P_{7eqm} + x_1)}{dt} = \frac{dx_1}{dt} = \alpha_1 x_1 + \alpha_2 x_4 + \alpha_3 u_1 + \alpha_4 u_2 \quad \text{(B.45)}$$

This is the first of the four scalar equations in Equation 5.35. Thus, $\alpha_1 = \boldsymbol{A}(1,1)$, $\alpha_2 = \boldsymbol{A}(1,4)$, $\alpha_3 = \boldsymbol{B}(1,1)$ and $\alpha_4 = \boldsymbol{B}(1,2)$. Other elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ in the first row are zero.

In this manner, all four model state derivative equations are derived to obtain the coefficients of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in the linearized model of Equation 5.34.

Glossary

Note: Unless otherwise indicated, variables are assumed dimensionless

Terms

b	damping constant	[N-s/m]
e	error	
C_p	specific heat capacity at constant pressure	[kJ/kg-K]
C	heat capacity	[kJ/K]
C_f	flow coefficient	[$kg^{0.5}/m^{0.5}$]
C_r	heat capacity ratio	
h	enthalpy	[kJ/kg]
HL	heat load	[kW]
J	objective function	
k	proportionality constant	
m	mass	[kg]
n	valve position	
P	pressure	[Pa]
q	rate of heat exchange	[kW]
s	Laplace variable	
s	entropy	[kJ/kg-K]
s	source term	[Pa]
t	time	[s]
T	temperature	[K]
u	model input	
W	mass flow rate	[kg/s]
x	model state	
x	quality	
ϵ	effectiveness	
η	efficiency	
ϱ	density	[kg/m^3]
τ	time constant	[1/s]
ω	rotational speed	[1/s]

Vectors and Matrices

A	state matrix
B	input matrix
K	feedback gain matrix
Q	state cost matrix
R	input cost matrix
S	Riccati matrix
U	model input vector
X	model state vector
$\dot{\mathbf{X}}$	model state derivative vector

Chiller Schematic Labels

1	compressor suction
2	compressor exit
3	condenser shell inlet
4	condenser shell exit
5	TEV suction
6	TEV exit
7	evaporator shell inlet
8	evaporator shell exit
9	evaporator tube inlet
10a	evaporator tube exit
10b	heat pipe exit
11	condenser tube inlet
12	condenser tube exit
HGBP	hot gas bypass valve exit

Generalized Subscripts

1ϕ	single-phase
2ϕ	two-phase
acutal	actual
co	condenser
cond	condenser
comp	compressor
d	derivative
des	design condition
e	evaporation
eqm	equilibrium
ev	evaporator
evap	evaporator
f	fluid
fw	fresh water
g	saturated vapor
HGBP	hot gas bypass valve
i	coefficient of linearized variation
i	integral
in	inlet
isen	isenthalpic
j	coefficient of linearized variation
k	coefficient of linearized variation
l	saturated liquid
lg	vaporization
m	metal
min	minimum
max	maximum
o	optimal
o	reference state
out	outlet/exit
p	proportional
pump	pump
sh	shell-side
sw	seawater
TEV	thermostatic expansion valve
tu	tube-side
v	vapor

References

[1] Zerby, M., "*Thermal Management for the Electric Warship*". Electric Machine Technology Symposium, 2006.

[2] Holsonback, C.R., "*Dynamic Thermal-Mechanical-Electrical Modeling of the Integrated Power System of a Notional All-Electric Naval Surface Ship*". Master's Thesis, The University of Texas at Austin, May 2007.

[3] "*ESRDC Background*". Electronic Ship Research and Development Consortium, 2007. <http://esrdc.caps.fsu.edu/esrdc.background.html>. June 24, 2012.

[4] "*ESRDC Consortium Thrusts*". Electronic Ship Research and Development Consortium, 2007. <http://esrdc.caps.fsu.edu/thrusts.html>. June 24, 2012.

[5] Carroll, B. C., "*Improved Thermal Management of an All-Electric Ship through Modeling and Simulation*". Master's Thesis, The University of Texas at Austin, 2004.

[6] Haag, S. T., "*Steady-State and Dynamic Simulation of Large Thermal Systems*". Master's Thesis, The University of Texas at Austin, December 2005.

[7] Sutter, H., *More exceptional C++: 40 new engineering puzzles, programming problems, and solutions*, Addison-Wesley, Boston, MA, 2002.

[8] Paullus, P. E., "*Creation of a Modeling and Simulation Environment for Thermal Management of an All-Electric Ship*". Master's Thesis, The University of Texas at Austin, December 2007.

[9] Hewlett, P. T., "*Implementation of an In-House Framework for Dynamic Assessment of Thermal Load Management Strategies aboard Navy Surface Ships*". Master's Thesis, The University of Texas at Austin, December 2008.

[10] Pruske, M. A., "*Thermal-Electrical Co-Simulation of Shipboard Integrated Power Systems on an All-Electric Ship*". Master's Thesis, The University of Texas at Austin, August 2009.

[11] He, X.-D., "*Dynamic Modeling and Multivariable Control Vapor Compression Cycles in Air Conditioning Systems*". PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, February 1996.

[12] Rasmussen, B. P., "*Dynamic Modeling and Advanced Control of Air Conditioning and Refrigeration Systems*". PhD Thesis, University of Illinois at Urbana- Champaign, Urbana, IL, August 2005.

[13] Llopis, R., Cabello, R., Navarro-Esbri, J. and Torella, E., *"A dynamic mathematical model of a shell-and-tube evaporator. Validation with pure and blend refrigerants"*. International Journal of Energy Research, Vol. 31, 2007, Pages 232-244.

[14] Llopis, R., Cabello, R., Torella, E., *"A dynamic model of a shell-and-tube condenser operating in a vapour compression refrigeration plant"*. International Journal of Thermal Sciences, Vol. 47, Issue 7, July 2008, pp. 926-934.

[15] Gruhle, W.-D., Iserman, R., *"Modeling and control of a refrigerant evaporator"*. Proceedings of the 1985 American Control Conference, June 1985, pp. 287-292.

[16] Shah, R., Alleyne, A. G., Bullard, C. W., Rasmussen, B. P., Hrnjak, P. S., *"Dynamic modeling and control of single and multi-evaporator subcritical vapor compression systems"*. Air Conditioning and Refrigeration Center, University of Illinois at Urbana-Champaign, Aug 2003.

[17] Wang, F. Q., Maidment, G. G., Missenden, J. F., Tozer, R. M., *"A novel special distribution method for dynamic refrigeration system simulation"*. International Journal of Refrigeration, Vol. 30, February 2007, pp. 887-903.

[18] Zhang, T., Catano, J., Rongliang, Z., Wen, J. T., *"Dynamic modeling of refrigeration cycle for electronics cooling"*. Proceedings of ASME International Mechanical Engineering Congress & Exposition, November 2008, Boston, USA.

[19] Naval Sea Systems Command, "*Air Conditioning Plant, HFC-236fa, 200-Ton Capacity (DDG-51 Thru DDG-82): Description, Operation, and Maintenance*". Technical Manual, March 2002.

[20] Lemmon, E.W., M.L. Huber, and M.O. McLinden, "*NIST Reference Fluid Thermodynamic and Transport Properties—REFPROP Version 8.0*". User's Guide and Technical Manual, Boulder, CO, April 2007.

[21] Thome, J.R., "*Wolverine Engineering Data Book III"*. Electronic resource, <http://www.wlv.com/products/databook/db3/DataBookIII.pdf>, Feb 2012.

[22] Serth, R.W., "*Process Heat Transfer"*. Elsevier Ltd., Oxford, UK, 2007.

[23] Incropera, F. P., DeWitt, D. P., "*Introduction to Heat Transfer"*. ISBN: 978-0471612476, John Wiley & Sons, Inc., Hoboken, NJ, 2001.

[24] Pierce, M., "*Dynamic Thermal Modeling and Simulation Framework: Design of Modeling Techniques and External Integration Tools*". Master's thesis, The University of Texas at Austin, December 2009.

[25] "*Cycle-Tempo*". Delft University of Technology, Netherlands <http://www.cycle-tempo.nl>, July 10, 2012.

[26] "*ProTRAX Overview | TRAX Energy Solutions*". TRAX International, Las Vegas, NV. <https://energy.traxintl.com/protrax/protrax-overview/>, July 10, 2012.

[27] Ogata, K., "*Modern Control Engineering*". Fourth Edition, ISBN: 978-0130609076, Pearson Education, Inc., Upper Saddle River, New Jersey, 2002.

[28] Messner, W., Tilbury, D., "*CTMS: PID Tutorial*". Electronic resource <http://www.engin.umich.edu/class/ctms/pid/pid.htm>, Jul 2012.

[29] Kautsky, J., Nichols, N.K., Van Dooren, P., "*Robust Pole Assignment in Linear State Feedback*". International Journal of Control, 41 (1985), pp. 1129-1155.

[30] Lewis, F. L., Syrmos, V. L., "*Optimal Control"*. ISBN: 0-471-03378-2, John Wiley & Sons, Inc., Hoboken, NJ, 1995.

[31] D'Azzo, J. J., Houpis, C. H., "*Linear Control System Analysis & Design*". ISBN: 0-07-016186-0, McGraw-Hill Book Company, The McGraw-Hill Companies, Columbus, OH, 1988.

[32] Browne, M. W., Bansal, P. K., "*Transient simulation of vapour-compression packaged liquid chillers*". International Journal of Refrigeration 25 (2002), pp. 597–610.

Vita

Gautam Salhotra was born and raised in Mumbai, India. He received his education from St. Stanislaus High School, Bandra (W), Mumbai, till the 10th grade and went to Kishinchand Chellaram College, Churchgate, Mumbai, for the remainder of his high school. Gautam went on to pursue a dual-degree in Mechanical Engineering from the Indian Institute of Technology Bombay, where he received his Bachelors and Masters of Technology in 2010. In the fall of 2010, he joined the University of Texas at Austin as a graduate student in the Mechanical Engineering Department, with a specialization in Dynamic Systems and Controls. In January 2011, Gautam started working on naval thermal management research at Applied Research Laboratories, supervised by Dr. Thomas Kiehne, and will be completing his graduate work in August 2012.

Contact information: gautam@utexas.edu

This thesis was typed by the author.